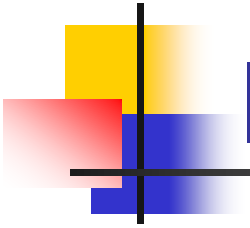
Programmation Web en PHP

Éléments du langage

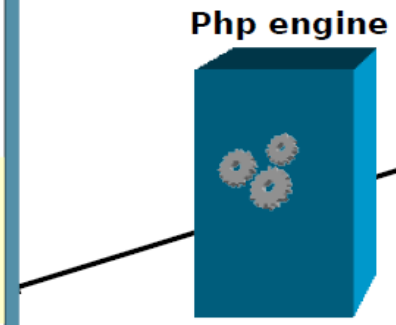


Introduction

- Définition
  - Personal Home Page ou (Hypertext PreProcessor)
    - Un langage de scripts évolué pour la conception de sites entiers :
      - ❖ s'intègre à HTML
    - Relativement simple à utiliser
      - ❖ fait notamment des miracles, couplé à un serveur de base de données
    - C'est un langage qui s'exécute du côté serveur
      - ➔ il est interprété par le serveur Web

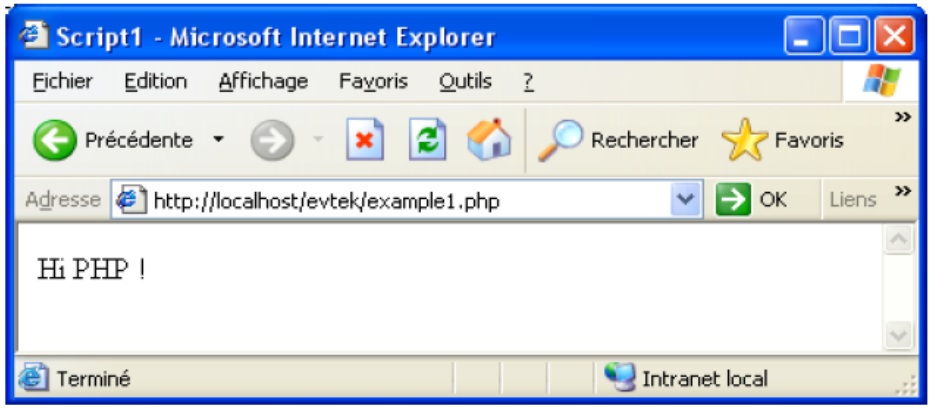
Exécution d'un script PHP

```
<html><head><title>
Script1
</title></head>
<body>
<?php  /* PHP code */
echo “Hi  PHP ! ";
?>
</body>
</html>
```



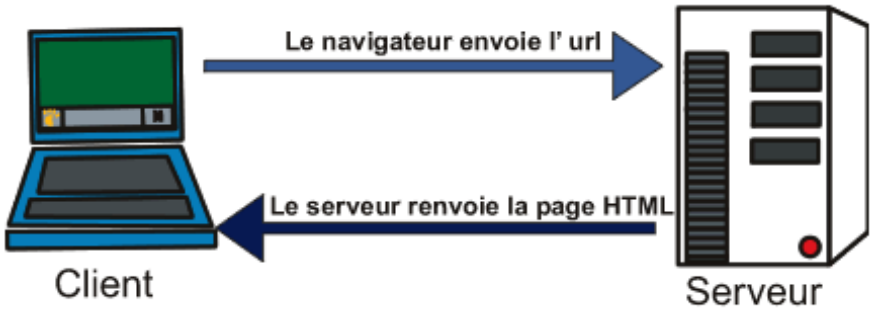

```
<html><head>
<title>Script1
</title></head>
<body>
Hi PHP !
</body>
</html>
```

Web server



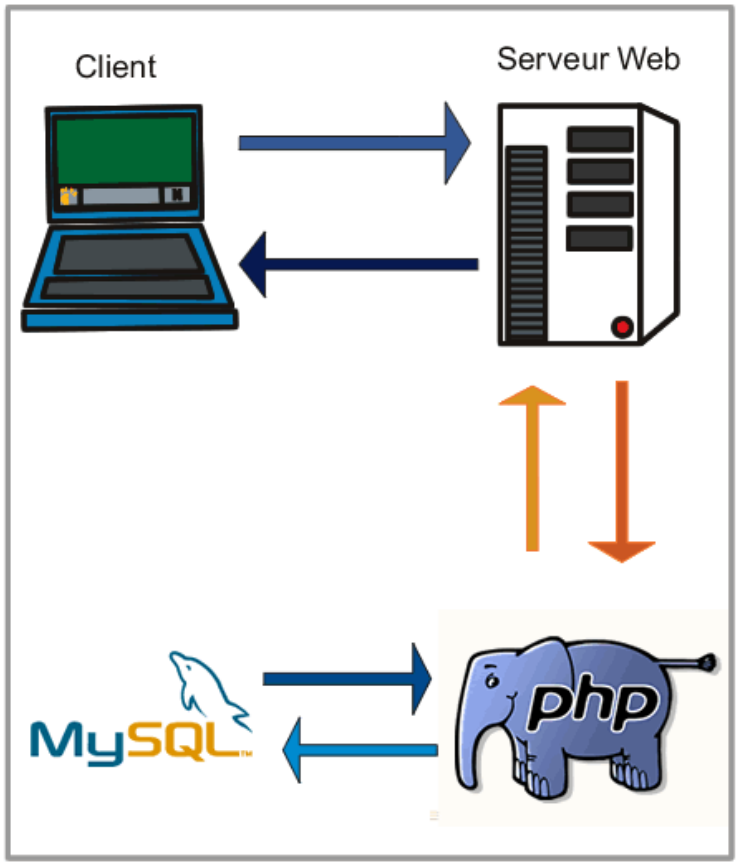
Introduction

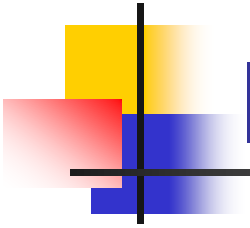
- Voici, en simplifiant, ce qui se passe lorsque vous consultez une page html
  - Le navigateur envoie l'adresse URL tapée
  - Le serveur web est un "ordinateur" présent sur l'Internet et qui héberge la page demandée
  - Sur ce serveur, on trouve Apache, logiciel apte à traiter les requêtes HTTP
  - Apache cherche le fichier demandé et renvoie à votre navigateur la page HTML
  - Votre navigateur interprète les différents langages se trouvant dans ce fichier (HTML, JavaScript, etc.) et affiche la page



Introduction

- Maintenant, voyons ce qui se passe lorsque votre page HTML contient du code PHP :
  - Le serveur regarde si le fichier envoyé contient une extension .php
  - Si oui, il transmet le fichier à PHP qui l'analyse et l'exécute
  - Si le code contient des requêtes vers MySQL, PHP envoie la requête SQL à MySQL
  - La base de données renvoie les informations voulues au script qui peut les exploiter
  - PHP continue d'analyser la page, puis retourne le fichier dépourvu du code PHP au serveur web
  - Le serveur web renvoie donc un fichier ne contenant plus de PHP, donc seulement du HTML au navigateur qui l'interprète et l'affiche





Introduction

- Utiliser PHP sur son ordinateur
  - Pourquoi installer PHP sur son ordinateur ?
    - Pour tester vos script PHP, vous allez être amenés à les envoyer sur votre hébergeur, sur Internet
    - Cependant il devient vite très lourd de sans cesse renvoyer ces fichiers par FTP
    - C'est pourquoi installer un serveur web sur son ordinateur est utile, et permet de tester ses scripts plus souplement
  - Concrètement, votre ordinateur sera à la fois client et serveur
    - Ainsi vous pourrez programmer en PHP sans avoir besoin d'être connecté à Internet, ce qui peut être utile pour les personnes ne disposant pas de connexions illimitées
  - Pour cela, il existe plusieurs utilitaires très pratiques qui installeront Apache
  - Le plus connu est : EasyPHP (Php4) : www.easyPHP.org

Première partie

Les structures de base

Les bases du PHP

- Créer un fichier php
  - Le code PHP est toujours encadré par des balises le signalant
  - Les balises possibles sont :
    - <?php ?>
    - ou <? ?>
    - ou <% %>
    - ou <script language="php"> </script>
  - Le fichier porte le suffixe .php

Les bases du PHP

Utilisation de EasyPhp

- Exemple : exemple0.php

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

  - Rangement du fichier

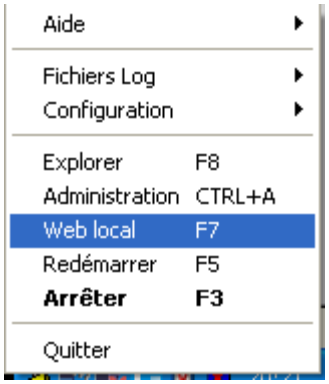
    Sur votre station :

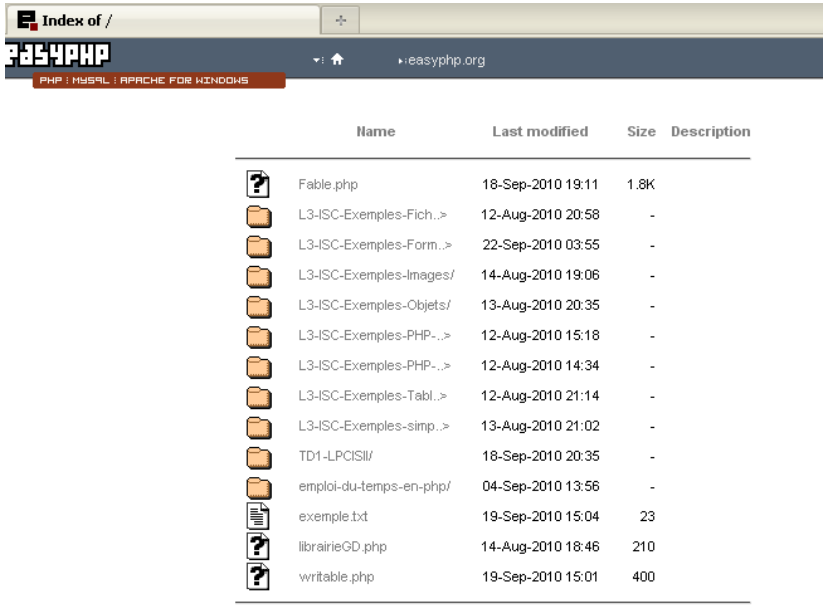
    C:\Program Files\EasyPHP1-8\www\exemple0.php

Les bases du PHP

Utilisation de EasyPhp

- Exécution :
  - Solution 1 : http://localhost/exemple0.php
  - Solution 2 : clic droit sur l'icône, puis clic sur Web local, vous trouverez le contenu de www



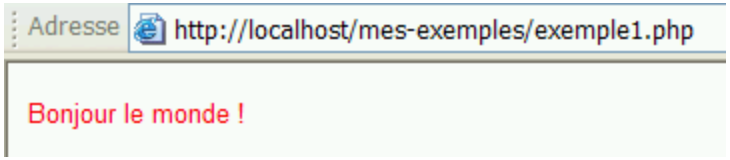


Les bases du PHP

- Du HTML dans du PHP : exemple1.php
  - echo permet d'introduire du code HTML
  - Exemple :

```
<?php
   echo '<font face="arial" size="2"
   color="red">Bonjour le monde !</font>';
?>
```

  - Résultat



- Nous avons ajouté la balise font en HTML pour formater le texte
- En fait PHP ne fait pas le formatage, il faut utiliser HTML pour ça

Les bases du PHP

- Autre exemple : exemple2.php
  - Affichage d'une image en plus du texte

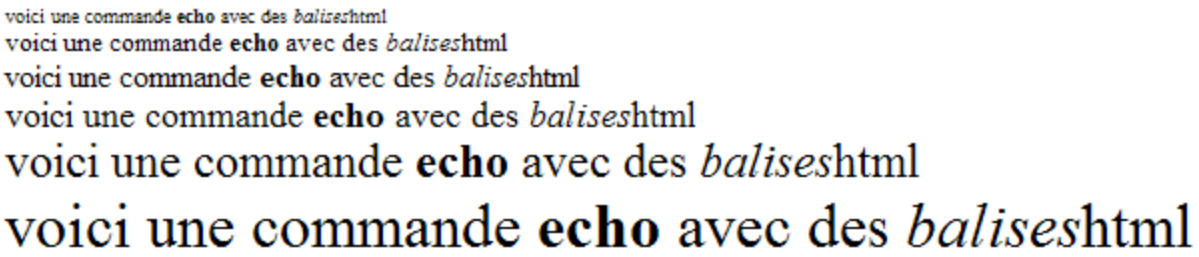
```
<?php
   echo '<div align="center"><font face="arial" size="2"
   color="blue"> Bonjour le monde !</font><br /> ';
echo '<img src="images/globe.gif" border="0" /></div> ';
?>
```

http://www.phpdebutant.org/article14.php

Les bases du PHP

- echo :
  - devient plus intéressante avec des variables :
    exemple3.php

```
<?php
   for ($i=1; $i<=6;$i++)
   {
            echo "<br>";
            echo "<font size= $i >";
            echo "voici une commande <b>echo</b> avec des
            <i>balises</i>html";
   }
?>
```



Les bases du PHP

- Du code PHP dans du HTML
  - A partir du moment où vous placez du code PHP dans un fichier *.htm ou *.html, vous devriez renommer ce fichier en *.php ou encore *.phtml
  - Le code php se place dans le body

Les bases du PHP

- Exemple : exemple4.php

```
<html>
<body>
  <font size="2" face="Arial">Le texte en HTML</font>
  // le code PHP ---------
  <?php
            $heure = date("H\hi"); //ex. 13h15
            //http://php.net/manual/fr/function.date.php
            print("<font size=\"2\" face=\"Arial\"> et celui en
            PHP.</font>");
            // on entoure \"2\" car 2 doit apparaître entre " "
   ?>
  <!--- retour au code HTML -->
  <br>
  <font size="2" face="Arial">Il est
  // de nouveau, du PHP ---------
  <?php echo $heure; ?>
  </font>
</body>
</html>
```

Les bases du PHP

- La fonction include : exemple5.php
  - Permet de ramener du code .php extérieur
  - Exemple

```
<html>
    <body>
        <font size="2" face="Arial">Le texte en HTML</font>
          <?php
              include("toto.inc.php"); // on appelle le
              // fichier toto.inc.php
          ?>
    </body>
</html>
```

Les bases du PHP

- Le code php de toto.inc.php est

```
<?php
  $heure = date("H\hi");
  print("<center><font size=\"2\" face=\"Arial\"> et celui
  en PHP. Il est $heure.</font></center>");
?>
```

Les bases du PHP

- La concaténation
  - Le point est utilisé pour concaténer des chaînes, des variables, etc.
  - Exemple
    - Phrase où un texte doit être collé au bout d'une variable
    - Le point après gmt permet d'indiquer à php la fin de la concaténation

```
<?
        $date = gmdate("H\hi");
        print("Il est $date"."gmt.");
?>
```

Les bases du PHP

- Variables dynamiques (suite)
  - Exemple : créer des variables par indiçage

```
<?php
   $v1 = "15 €";
   $v2 = "30 €";
   $v3 = "dvd";
   for($i=1;$i<=3;$i++)
    echo ${"v".$i}."<br/>";
?>
```

Les bases du PHP

- Constantes et variables
  - Constantes : constante.php
    - On les définit à l'aide de la fonction define()

```
<?php
   define("NOM", "Anaska");
   echo NOM;
?>
//écrit Anaska
```

Les bases du PHP

- Interprétation des variables
  - À l'intérieur d'une chaîne entre guillemets, les variables sont automatiquement remplacées par leur valeur
  - Exemple : interpretation.php

```
<?php
   $objet = "livre";
   $chaine ="Son $objet a déclenché la légende";
   echo $chaine;
   //Affiche Son livre a déclenché la légende
?>
```

Structures de contrôle

- Les conditions
  - Première forme

```
If(condition){
    //instructions
}
```

  - Deuxième forme

```
If(condition){
    //instructions
}else{
    \\instructions
}
```

  - Troisième forme

```
If(condition){
    //instructions
}elseif{
    //instructions
}else{
    //instructions
}
```

Structures de contrôle

- Les conditions (suite)
  - Quatrième forme : switch

```
<?php
    $nombre = mt_rand(0,4) //génère un nombre aléatoire entre 0 et 4
    switch ($nombre){
    case 4 :
            echo "$nombre est supérieur à 3<br>";
    case 3 :
            echo "$nombre est supérieur à 2<br>";
    case 2 :
            echo "$nombre est supérieur à 1<br>";
    case 1 :
            echo "$nombre est supérieur à 0<br>";
    default :
            echo "$nombre est 0<br>";
}
?>
```

Structures de contrôle

- Exemple : if then else
  -
```
<?php
  if( $var == 'ok')
  {
            print 'test';
  }
  else{
            print 'refusé';
  }
?>
```

- Exemple : if else elseif

```
$variable = 'voiture';

  if($variable == 'voiture'){
    print 'bravo vous avez
    trouvé';
  }
  elseif($variable=='automobile'){
    print 'c\'est presque ça';
  }
  else {
    print 'ce n\'est pas ça veuillez
    réessayer';
  }
```

Structures de contrôle

Exemple : switch()

```
switch($operation)
{
    case '1': // si la variable opération est égale à 1
      print ' opération numero 1'; // on affiche cette phrase
      break; // on referme cette condition

    case '2': // si la variable opération est égale à 2
      print 'opération numero 2';
      break;

    default: // si la variable opération n' est pas égale à 1 ni à 2
    ou si elle n'est pas définie
      print 'opération par defaut'; // on affiche une phrase par
      défaut
    }
```

Structures de contrôle

- Les conditions multiples
  - permettent de donner plusieurs conditions pour effectuer une ou plusieurs actions

```
<?php
        $homme = FALSE; //booléen ayant la valeur FALSE (faux)
              ici il s'agit donc d'une femme
        $age   = 17;

        if($homme === TRUE AND $age > 13) //Le visiteur est un
              homme et âgé de plus de 13 ans
        {
             echo 'Vous pouvez visiter le site';
        }
        else //Le visiteur est une femme ou alors il a moins de 13 a
              ns
        {
             echo 'Vous ne pouvez pas visiter le site';
        }
?>
```

Structures de contrôle

- Les conditions multiples
  - Autre exemple : importance des parenthèses

```
<?php
        if($homme === TRUE OR ($homme === FALSE
            AND $age > 13)) //On veut soit tous les
            hommes, soit les filles de plus de 13 ans
        {
            echo 'Vous pouvez visiter le site';
        }
        else
        {
            echo 'Vous ne pouvez pas visiter le site';
        }
?>
```

Structures de contrôle

- Les conditions multiples
  - On peut utiliser les opérateurs prioritaires && et || pour supprimer les parenthèses
  - Voici le code obtenu :

```
<?php
        if($homme === TRUE OR $homme === FALSE && $ag
              e > 13) //On veut soit tous les hommes, soit les filles d
              e plus de 13 ans
        {
             echo 'Vous pouvez visiter le site';
        }
        else
        {
             echo 'Vous ne pouvez pas visiter le site';
        }
?>
```

Structures de contrôle

- Les conditions multiples
  - Comme le && est prioritaire, PHP effectue d'abord le traitement pour savoir si il s'agit d'une fille ayant plus de treize ans
  - On pourrait simuler ça par ce code :

```
<?php
        if($homme === TRUE OR $fille_de_plus_de_treize_ans =
              == TRUE) //On veut soit tous les hommes, soit les fille
              s de plus de 13 ans
        {
             echo 'Vous pouvez visiter le site';
        }
        else
        {
             echo 'Vous ne pouvez pas visiter le site';
        }
?>
```

  - Ensuite PHP utilise le OR classique pour faire une condition entre les deux variables.

Structures de contrôle

- L’opérateur ===
  - Cet opérateur permet de valider une condition si les variables ont même valeur et même type
  - En fait, un booléen peut aussi être représenté par un nombre (0 pour FALSE et 1 pour TRUE)
  - Le problème est que lorsque vous utiliserez des fonctions qui renvoient des booléens ou des nombres, comment distinguer 0 et 1 de FALSE et TRUE ?
  - C'est là qu'intervient le signe ===, qui vous permettra de savoir si la fonction a renvoyé TRUE ou 1, ce que ne permet pas de faire l'opérateur ==

Structures de contrôle

- Les boucles
  - Première forme

```
while(condition){
    //instructions
}
```

  - Deuxième forme

```
do {
    //instructions
}
while (condition)
```

  - Troisième forme

```
for
(expression1;condition;expres
sion2){
//instructions
}
```

Structures de contrôle

- Exemple avec while()

```
$i= 0; // on défini une variable à 0 pour le compteur de boucle
while ( $i <= 4 ) // la boucle s'arrêtera lorsque la variable $i sera égale à 4
{
    print 'boucle numero '.$i.'<br />'; // on affiche une phrase avec le numero de la boucle
    $i++; // le ++ sert à ajouter 1 à chaque tour de boucle, ne l'oubliez pas sinon la boucle sera infini donc affichera une erreur !
}
```

- Affichera à l'écran

boucle numero 0
boucle numero 1
boucle numero 2
boucle numero 3
boucle numero 4

Structures de contrôle

- Les fonctions utilisateur
  - Déclaration

```
<?php
    function Nom_de_la-fonction($argument1, $argument2,
    …){
    //liste d'instructions
}
?>
```

  - Valeur par défaut

```
<?php
    function Nom_de_la-fonction($argument1='valeur_par_defaut'){
    //liste d'instructions
}
?>
```

  - Valeur de retour
    - La fonction peut renvoyer une valeur grâce au mot-clé : return
    - Une fonction peut contenir plusieurs instructions de retour, mais l'exécution s'arrêtera à la première mise en oeuvre

Structures de contrôle

- Les fonctions utilisateur
  - Exemple : fonction-return.php

```
<?php
function dire_texte($qui, $texte='Bonjour'){
    if(empty($qui)){
     return FALSE;
    }else{
        echo "$texte $qui";
        return TRUE;
    }
}
```

Structures de contrôle

- Appel
  - Nom_de_la_fonction(argument1, argument2, …)
- Exemple :

```
<?php
function dire_texte($qui, $texte='Bonjour'){
    if(empty($qui)){
     return FALSE;
    }else{
        echo "$texte $qui";
        return TRUE;
    }
}
dire_texte('cher phpeur', 'bienvenue');
//Utilisation de la valeur par défaut
dire_texte('cher phpeur');
?>
```

Structures de contrôle

- Appel
  - On peut aussi contrôler le retour

```
<?php
function dire_texte($qui, $texte='Bonjour'){
    if(empty($qui)){
     return FALSE;
    }else{
        echo "$texte $qui";
        return TRUE;
    }
}
if (dire_texte(""){
    echo "Erreur";
};
if (!dire_texte("cher phpeur")
//Affiche "Bonjour cher phpeur"
?>
```

Structures de contrôle

- Les fonctions utilisateur
  - Passage de paramètres par recopie
    - Par défaut, PHP fait un passage par recopie
    - La valeur utilisée par la fonction n'est donc pas celle donnée en argument mais une copie
      - Si vous la modifiez à l'intérieur de la fonction, cela n'aura pas d'influence dehors
  - Exemple

```
<?php
        function ajouter_cinq($nombre)
        {
            $nombre += 5; //équivalent de $nombre = $nombre + 5
            return $nombre;
        }

        $mon_entier = 15;
        echo ajouter_cinq($mon_entier); //affichera 20

        echo $mon_entier; //affichera 15
?>
```

Structures de contrôle

- Les fonctions utilisateur
  - Passage de paramètres par référence
    - On fait référence à la variable dans le programme appelant et tout ce qu'on fait sur la variable est reoportée au niveau du programme appelant
    - Pour cela, il faut accompagner le paramètre d'appel de "&"
  - Exemple

```
<?php
        function ajouter_cinq($nombre)
        {
            $nombre += 5; //équivalent de $nombre = $nombre + 5
            return $nombre;
        }

        $mon_entier = 15;
        echo ajouter_cinq(&$mon_entier); //affichera 20

        echo $mon_entier; //affichera 20
?>
```

Structures de contrôle

- Passage par référence (suite)
  - L'avantage de ce type d'opération est que vous travaillez directement sur la variable d'origine, il n'y a pas de recopie et donc les performances peuvent être meilleures
  - Vous n'avez d'ailleurs plus forcément besoin de retourner une valeur
  - Prenons cet exemple qui fait exactement la même chose que le précédent :

```
<?php
        function ajouter_cinq($nombre)
        {
            $nombre += 5; //équivalent de $nombre = $nombre + 5
        }

        $mon_entier = 15;
        ajouter_cinq(&$mon_entier);

        echo $mon_entier; //affichera 20
?>
```

Les tableaux

- Généralités
  - Déclaration : plusieurs manières :

```
<?php
    // Déclaration d'un tableau vide
$fruits = array();
    // Déclaration d'un tableau indexé numériquement
$legumes =
array('carotte','poivron','aubergine','chou');
    // Déclaration d'un tableau mélangeant les types
entier et chaine
$tab = array($variable, “texte”, 153, 56);
?>
```

Les tableaux à indices numériques

- Déclaration
  - En PHP, la déclaration est implicite, nul besoin de préciser à l'avance le nombre d'éléments du tableau...
  - Par affectation

```
$t[0]="bonjour";
$t[1]="bonsoir";
$t[2]="bla bla bla";
```

  - Utilisation

```
echo "case numéro 2 : ".$t[2]."<BR>\n";
for ($i=2 ; $i<6 ; $i++) {
  echo "case numéro $i : ".$t[$i]."<BR>\n";
}
```

Les tableaux associatifs

- Création
  - Pour créer un tableau associatif, il faut donner pour chaque élément, le couple : (clé => valeur)

```
<?php
   $tab = array(
            "prenom" => "Cyril";
             "ville" => "Paris";
            "travail" => "informatique"
   );
?>
```

Clé	Valeur
prenom	Cyril
ville	Paris
travail	informatique

Les tableaux associatifs

- Exemples de tableaux simples :
  - clé => valeur

```
$fruits  = array ("a"=>"orange", "b"=>"banane", "c"=>"pomme");
$trous = array (1=>"premier", 2 => "second", 3 => "troisième");
```

- Exemple de tableau de tableaux :
  - clé =N° de département => sous-tableau :
  - Chaque sous- tableau est composé de 4 éléments : région, nom du département, et coordonnées (4 nombres) sur une carte de France

```
var $departement = array (
   "01"  => array ( "Rhône Alpes","Ain","236","222","255","243" ),
   "02"  => array ( "Picardie","Aisne","192","97","209","122" ),
   "03"  => array ( "Auvergne","Allier","176","215","201","232" ),
   etc.
```

Les tableaux associatifs

- Fonctions relatives : isset
  - Pour tester l'existence d'un élément, on utilise la fonction isset()
  - Exemple :

```
$calories["pommes"]= 300;
$calories["banane"]= 130;
$calories["litchi"]= 30;

if( isset( $calories["pommes"] ) ) {
   echo "une pomme contient ", $calories["pommes"] ,
   "calories";
}
else{
   echo "pas de calories définies pour la pomme";
}
```

Les tableaux associatifs

- Parcours :
  - La méthode classique ne fonctionne pas. Il faut utiliser les fonctions : foreach, list…
  - Exemple : parcours-tableau-assoc.php

```
<?php
        $tableau = array(
            0 => 'Chiffre 0',
            1 => 'Chiffre 1',
            2 => 'Chiffre 2‘
        );
        foreach($tableau AS $cle => $valeur) {
                echo $cle.' | '.$valeur.'<br />';
        }
?>
```

Parcours : parcous-tab-assoc3.php

```
<html>
  <body>
   <?php
    $annee_modif = "";
     $mois_modif = array("January" => "Janvier", "February" => "Février",
     "March" => "Mars", "April" => "Avril", "May" =>    "Mai",
     "June" => "Juin", "July" => "Juillet", "August" => "Août", "September" =>
     "Septembre", "October" => "Octobre", "November" => "Novembre",
     "December" => "Décembre");


     $date_modif = date( "d F Y", getlastmod());
      list($jour, $mois, $annee) = preg_split( '[ ]', $date_modif);
      foreach($mois_modif as $cMM => $MM)
                  if($cMM == $mois) $mois = $MM;
     $date_modif = "Dernière modification : $jour $mois $annee";
     echo $date_modif;
   ?>
  </body>
</html>
```

Utilisation des tableaux

- Rechercher un élément
  - Présence d'un élément :
    - in_array(expression, tableau)
  - Exemple

```
<?php
   $colors = array('rouge', 'vert', 'bleu');
   if (in_array('vert', $colors)){
    echo '<br>Trouvé, vert';
}
?>
```

Utilisation des tableaux

- Calculer la clé :
  - Array_search(expression, tableau)
  - Exemple

```
<?php
   $colors = array('rouge', 'vert', 'bleu');
   $cle = array_search('vert', $colors);
    echo "La valeur 'vert' est à la clé $cle";
   //Affiche : la valeur 'vert' est à la clé 1
?>
```

Utilisation des tableaux

- Vérifier l'existence d'une clé
  - array_key_exists()
  - Exemple

```
<?php
   $colors = array('ff0000' => 'rouge', '00ff00' => 'vert',
   '0000ff' => 'bleu');
   if(array_key_exists('00ff00', $colors)){
            echo 'La clé "00ff00" existe';
   }
?>
```

Utilisation des tableaux

- Calculer le nombre d'occurrences d'un élément
  - array_count_values()
  - Exemple

```
<?php
   $tab = array('Cyril', 'Christophe', 'Cyril' , 'Thomas', 'Eric');
   //tableau contenant les décomptes des éléments
    $cpt = array_count_values($tab);
   echo "L'élément 'Cyril' apparaît ", $cpt['Cyril'],
   "fois.<br>";
?>
```

Utilisation des tableaux

- Extraire et remplacer un élément
  - Par utilisation de : list()
  - Exemple

```
<?php
   $tab = array(1, 2, 3, 4);
   list($a, $b, $c, $d) = $tab;
   echo "$a-$b-$c-$d";
?>
```

➔ affiche : 1-2-3-4

Utilisation des tableaux

- Extraire des indices
  - extract() permet de faire des clés, des variables, et de leur donner la valeur de leur indice

```
<?php
   $tab = array(‘a’=>1, ‘b’=> 2, ‘c’=> 3, ‘d’=> 4);
   extract($tab);
   echo "$a-$b-$c-$d";
?>
```

➔ affiche 1-2-3-4

Utilisation des tableaux

- Gérer les clés utilisées
  - Liste des clés utilisées
    - array_keys()
  - Exemple

```
<?php
   $tab = array(‘a’=>1, ‘c’=> 5);
   $cles = array_keys($tab);
   echo implode(‘-’, $cles);
   //Affiche a-c
?>
```

Utilisation des tableaux

- Fusionner et séparer
  - Fusion de plusieurs tableaux
    - array_merge()
  - Exemple

```
<?php
    $result_2002 = array(12250, 12000, 21300, 25252, 20010, 8460);
    $result_2003 = array(1520, 25000, 13530, 1052, 5010, 3680);

    $result_2002_2003 = array_merge($result_2002, $result_2003);
    print_r($result_2002_2003);
?>
```

  - Affiche :

```
Array ( [0] => 12250 [1] => 12000 [2] => 21300 [3] => 25252 [4] =>
    20010 [5] => 8460 [6] => 1520 [7] => 25000 [8] => 13530 [9] =>
    1052 [10] => 5010 [11] => 3680 )
```

Utilisation des tableaux

- Séparer
  - array_chunk($tab,n)
    - sépare $tab en tableaux de n éléments chacun
- Calculer des différences et des intersections
  - Différence : array_diff
  - Exemple

```
<?php
    $tab1 = array(1, 2, 3, 4, 5, 6, 7);
    $tab2 = array(1, 3, 5, 7);
    $tab3 = array(1, 2, 3);

    $diff = array_diff($tab1,$tab2,$tab3);
    echo implode('-', $diff);
    //Affiche 4-6
?>
```

Utilisation des tableaux

- intersection : array_intersect()
- Exemple

```
<?php
    $tab1 = array(1, 2, 3, 4, 5, 6, 7);
    $tab2 = array(1, 3, 5, 7);
    $tab3 = array(1, 2, 3);

    $inter = array_intersect($tab1,$tab2,$tab3);
    echo implode('-', $diff);
    //Affiche 1-3
?>
```

Utilisation des tableaux

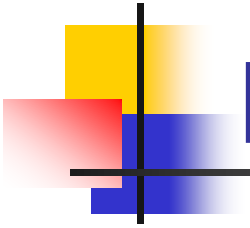
- Gérer des piles et des files
  - Fonctions : array_push et array_pop
    - array_push

```
<?php
    $tab = array();
    array_push($tab,1, 3, 5);
    /*equivalent à */
    $tab = array();
    $tab[1]=1;
    $tab[2]=3;
    $tab[3]=5;
?>
```

Utilisation des tableaux

- Fonctions : array_push et array_pop
  - array_pop

```
<?php
    $tab = array();
    array_push($tab,1, 3, 5);
    echo array_pop($tab); //Affiche 5;
    echo array_pop($tab); //Affiche 3;
?>
```



Les formulaires

- Intérêt
  - Dans un contexte Web, les données échangées avec le système se font à travers des formulaires
  - Les formulaires HTML sont la méthode la plus simple pour avoir des interactions avancées avec l'utilisateur
  - Ils permettent, par exemple, de :
    - Créer un espace sécurisé
    - Donner aux clients la possibilité de modifier eux-mêmes leurs sites
    - Interagir avec le visiteur en lui demandant des informations complémentaires…

Les formulaires

- Création : balise <form>

```
<form action='reception_formulaire.php' method='GET ou POST '>
<!-- différents champs -->
</form>
```

  - action :
    - désigne la page vers laquelle seront envoyées les informations rentrées par l'utilisateur une fois le bouton d'envoi actionné
  - method
    - définit le mode d'envoi des informations au serveur
    - Deux méthodes existent
      - GET et POST
    - PHP associe à ces deux variables deux tableaux $_GET et $_POST pour récupérer les données passées

Les formulaires

- Création : méthode POST
  - Exemple

```
<form action="valider.php" method="post">
  <select name="objet">
  ...
  <input name="genre" type="text" />
  …
```

  - Ce code HTML spécifie que les données du formulaire seront soumises à la page web “valider.php” en utilisant la **méthode POST**
  - Prenez soin de noter les noms (après name) des données du formulaire, car ils représentent les "clés" dans le tableau associatif "$_POST"
  - Exemple : **$_POST[‘genre']** permettra de récupérer genre
  - **Ces variables seront cachées pendant l’envoi**

Les formulaires

- Création : méthode GET (transmission par URL)
  - La différence avec la méthode POST est qu'elle passe les variables à la page web "valider.php" en les ajoutant à la fin de l'URL
  - Après avoir cliqué soumettre, l'URL aura ceci ajouté à la fin :
    - "valider.php?objet=xxx&genre=xxx"
  - Le point d'interrogation "?" dit au navigateur que les **objets suivants sont des variables**
  - **On les récupérera en utilisant le tableau $_GET[]**
  - **Dans ce mode de transmission, les variables sont apparentes**
  - **Pour votre utilisation c'est égal**

Les formulaires

Les éléments du formulaire

- Champ de texte

```
<input type="text" name="pseudo" value="M@teo21" />
```

  - 2 attributs
    - **name** : c'est le nom du champ de texte
      - Choisissez-le bien, car c'est lui qui va produire une variable
        - $_POST['pseudo']
    - **value** : permet d'affecter une valeur à la zone de texte
      - Par défaut, le champ de texte est vide

Les formulaires

Les éléments du formulaire

- Champ de texte : tester l'exemple : form-texte1.php

```
<?php
  if (isset($_POST['mon_champ'])) {
?>
   Votre champ contenait :
   <b> <?php echo $_POST['mon_champ']; ?></b>
   <br/><br/>
<?php
}
?>
<form method="POST">
   <input name="mon_champ" type="text"/>
   <input name="valider" type="submit" value="OK"/>
</form>
```

  - La fonction isset() permet de savoir si la variable *$_POST['mon_champ']* existe

- Champ de texte : autre manière : form-texte2.php

```
<?php
$mon_champ = isset($_POST['mon_champ']) ?
    $_POST['mon_champ'] : '';
      // ... dont la forme équivalente avec des if/else est :
      //if (isset($_POST['mon_champ'])) {
      //    $mon_champ = $_POST['mon_champ'];
      //} else {
      //    $mon_champ = '';
      //}
if ($mon_champ) { //ici, on a créé une variable pour le test
?>
    Votre champ contenait :
    <b><?php echo $mon_champ; ?></b>
    <br/><br/>
<?php
}
?>
<form method="POST">
    <input name="mon_champ" type="text" value=" <?php echo
    $mon_champ; ?>"/>
    <input type="submit" value="OK"/>
</form>
```

Les formulaires

Les éléments du formulaire

- Zone de texte : textarea

```
<textarea name="message" rows="8" cols="45">
        Votre message ici.
</textarea>
```

  - Remarques
    - rows resp. cols détermine le nombre de lignes resp. le nombre de colonnes de la zone de texte
    - Il n'y a pas d'attribut value
    - En fait, le texte par défaut est ici écrit entre le <textarea> et le </textarea>

Les formulaires

Les éléments du formulaire

- Exemple 1 : form-textarea1.php
  - Afficher le contenu de la zone de texte si celle-ci n'est pas vide

```
<?php
$mon_champ = isset($_POST['mon_champ']) ?
    $_POST['mon_champ'] : '';

if ($mon_champ) {
?>
    Votre champ contenait :
    <b><?php echo $mon_champ; ?></b>
    <br/><br/>
<?php
}
?>

<form method="POST">
    <textarea name="mon_champ"></textarea>
    <input type="submit" value="OK"/>
</form>
```

Les formulaires

Les éléments du formulaire

- Les boutons d'options (radio)
  - Permettent de faire des choix

```
Aimez-vous les frites ?
<input type="radio" name="frites" value="oui"
   checked="checked" /> Oui
<input type="radio" name="frites" value="non" /> Non
```

  - On utilise le même nom pour la gestion de tous les boutons, ici : « frites »
  - C'est donc cette variable qui contiendra ce que l'utilisateur a choisi
  - Dans la page cible, une variable $_POST['frites'] sera créée
  - Elle aura la valeur du bouton d'option choisi par le visiteur
  - Si on aime les frites, alors on aura $_POST['frites'] = 'oui'
  - Il faut bien penser à remplir l'attribut "value" du bouton d'option car c'est lui qui va déterminer la valeur de la variable

Les formulaires

Les éléments du formulaire

- Exemple : form-bouton-option1.php

```
<?php
    $mon_champ = isset($_POST['mon_champ']) ? $_POST['mon_champ'] : '';
    if ($mon_champ) {
?>
   Vous avez choisi :
   <b><?php echo $mon_champ; ?></b>
   <br/><br/>
<?php
}
?>
//on utilise la même variable mon_champ
<form method="POST">
   <input type="radio" name="mon_champ" value="Option 1"/>Option 1<br/>
   <input type="radio" name="mon_champ" value="Option 2"/>Option 2<br/>
   <input type="radio" name="mon_champ" value="Option 3"/>Option 3<br/>
   <input type="submit" value="OK"/>
</form>
```

Les formulaires

Les éléments du formulaire

- Les cases à cocher
  - Ici, on fait appel à un tableau pour retenir toutes les cases cochées
  - Exemple : form_case-a-cocher1.php

```
<?php
if (isset($_POST['mon_champ'])) {
    echo "Vous avez choisi :";
    for ($i = 0, $c = count($_POST['mon_champ']); $i < $c; $i++) {
        echo "<br/><b>" . $_POST['mon_champ'][$i] . "</b>";
    }
}
?>
<form method="POST">
    <input type="checkbox" name="mon_champ[]" value="Option 1"/>Option 1<br>
    <input type="checkbox" name="mon_champ[]" value="Option 2"/>Option 2<br>
    <input type="checkbox" name="mon_champ[]" value="Option 3"/>Option 3<br>
    <input type="submit" value="OK">
</form>
```

Les formulaires

Les éléments du formulaire

- Commentaire
  - La variable **$_POST['mon_champ']** est un tableau qui contient les valeurs que l'utilisateur a cochées
  - Pour signifier que les cases cochées seront des éléments d'un tableau, nous avons donc ajouté des crochets

Les formulaires

Les éléments du formulaire

- Cases à cocher (suite)
  - Dans l'exemple suivant, nous allons réafficher dans le formulaire les cases sélectionnées par l'utilisateur
  - Encore une fois, nous pourrions utiliser un autre tableau qui contiendrait toutes les cases du formulaire, ce qui nous permettrait d'afficher la case cochée ou non dans une simple boucle foreach mais nous allons rester au plus simple pour que tout ceci soit plus clair !

```php
<?php
if (isset($_POST['mon_champ'])) {
    echo "Vous avez choisi :";
    for ($i = 0, $c = count($_POST['mon_champ']); $i < $c; $i++) {
        echo "<br/><b>" . $_POST['mon_champ'][$i] . "</b>";
    }
}
// Renvoie vrai si $option fait partie du résultat
function est_selectionne($option) {
    if (!isset($_POST['mon_champ'])) {
            return FALSE;
    }
    for ($i = 0, $c = count($_POST['mon_champ']); $i < $c; $i++) {
        if ($_POST['mon_champ'][$i] == $option) {
            return TRUE;
        }
    }
    return FALSE;
}
?>
<form method="POST">
    <input type="checkbox" name="mon_champ[]" value="Option 1" <?php
      if(est_selectionne("Option 1")) { echo 'checked'; } ?>/>Option 1<br/>
    <input type="checkbox" name="mon_champ[]" value="Option 2" <?php
      if(est_selectionne("Option 2")) { echo 'checked'; } ?>/>Option 2<br/>
    <input type="checkbox" name="mon_champ[]" value="Option 3" <?php
      if(est_selectionne("Option 3")) { echo 'checked'; } ?>/>Option 3<br/>
    <input type="submit" value="OK"/>
</form>
```

http://www.siteduzero.com/tutoriel-3-14543-php-et-les-formulaires.html#ss_part_1

Les formulaires

Les éléments du formulaire

- Les listes déroulantes à sélection simple
  - Les listes déroulantes sont couramment utilisées pour que l'utilisateur ne puisse sélectionner qu'une valeur

```
<select name="mon_champ">
    <option>Option 1</option>
    <option>Option 2</option>
    <option>Option 3</option>
</select>
```

Les formulaires

Les éléments du formulaire

- Exemple : form-liste-deroulante1.php

```
<?php
$mon_champ = isset($_POST['mon_champ']) ? $_POST['mon_champ'] : '';
if ($mon_champ) {
?>
    Votre champ contenait :
    <b><?php echo $mon_champ; ?></b>
    <br/><br/>
<?php
}
?>
<form method="POST">
    <select name="mon_champ">
        <option>Option 1</option>
        <option>Option 2</option>
        <option>Option 3</option>
    </select>
    <input type="submit" value="OK"/>
</form>
```

Les formulaires

Les éléments du formulaire

- Liste déroulante à sélection multiple
  - Les listes déroulantes peuvent également être utilisées pour permettre de sélectionner plusieurs valeurs (voire une ou pas du tout)
  - Ce cas est strictement identique aux cases à cocher puisque l'on récupérera au final une variable "tableau" qui hébergera le résultat
  - Voyons un premier exemple avec un simple réaffichage des valeurs sélectionnées par l'utilisateur :

Exemple 1 : form-liste-deroulante3.php

```php
<?php
// Nos options définies dans un tableau (plus facile à coder et à maintenir)
$options = array(
    'Option 1',
    'Option 2',
    'Option 3',
    'Option 4'
);
// Affichage de la sélection seulement si le formulaire a été validé
if (isset($_POST['mon_champ'])) {
    echo "Vous avez choisi :";
    for ($i = 0, $c = count($_POST['mon_champ']); $i < $c; $i++) {
        echo '<br/><b>' . $_POST['mon_champ'][$i] . '</b>';
    }
}
?>
<form method="POST">
    <select name="mon_champ[]" size="4" multiple>
        <?php
        foreach ($options as $k) {
            echo '<option>' . $k . '</option>';
        }
        ?>
    </select>
    <input type="submit" value="OK"/>
</form>
```

http://www.siteduzero.com/tutoriel-3-14543-php-et-les-formulaires.html#ss_part_1

Les formulaires

Les éléments du formulaire

- Les champs cachés
  - En quoi ça consiste ?
    - C'est un code dans votre formulaire qui n'apparaîtra pas aux yeux du visiteur, mais qui va quand même créer une variable avec une valeur
    - Supposons que vous ayez besoin de "retenir" que le pseudo du visiteur est "Mateo21"
    - Vous allez taper ce code :
      - Code : HTML1<input type="hidden" name="pseudo" value="Mateo21" />
    - A l'écran, vous ne verrez rien
    - Mais dans la page cible, une variable $_POST['pseudo'] sera créée (correspondant à name), et elle aura la valeur "Mateo21" (correspondant à value) !
    - C'est apparemment inutile, mais vous verrez que lorsque vous commencerez à créer des formulaires vous en aurez vite besoin

Les formulaires

Les éléments du formulaire

- Exemple
  - Voici un exemple en HTML :

```
<form action="traitement.php" method="post">
<p><input type="hidden" name="champ_cache" value="ici ce
   que vous voulez" />Oui</p>
<p><input type="submit" value="Envoyer" /></p>
</form>
```

  - Et voici comment on récupère la valeur de ce champ caché en PHP :

```
<?php
    if(isset($_POST['champ_cache']))
    {
        echo htmlentities($_POST['champ_cache']); //affiche le
        contenu du champ
    }
?>
```