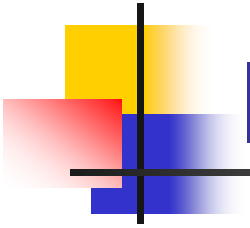
Programmation Web en PHP

Structures de base

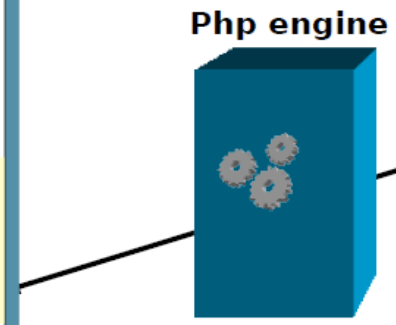


Introduction

- 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

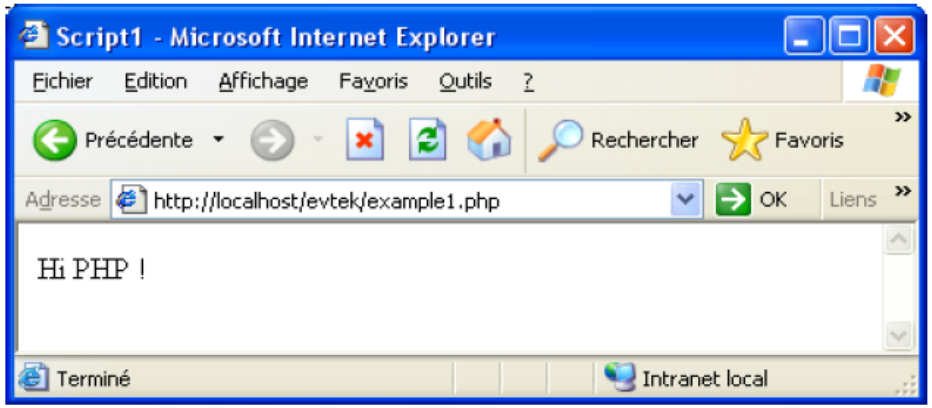
Exécution d'un script PHP

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



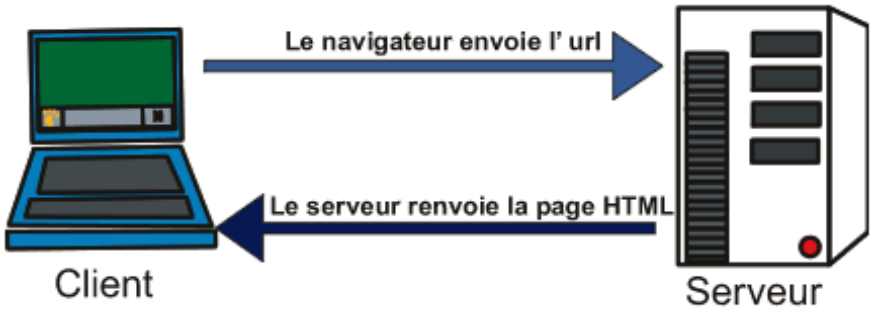

```
<html><head>
<title>Script1
</title></head>
<body>
Hi PHP !
</body>
</html>
```

Web server



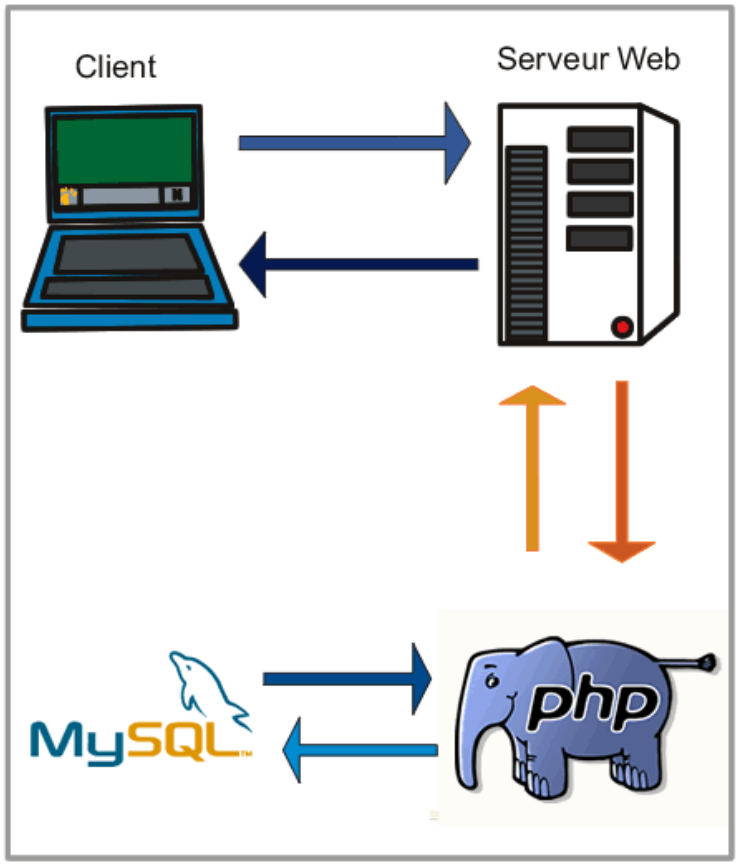
Introduction

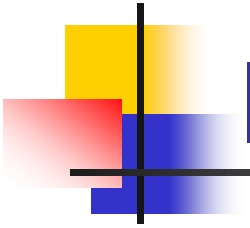
- Voici, en simplifiant, ce qui se passe lorsque vous consultez une page html
  - Le navigateur envoie l'adresse URL tapée
  - Le serveur web est un "ordinateur" présent sur l'Internet et qui héberge la page demandée
  - Sur ce serveur, on trouve Apache, logiciel apte à traiter les requêtes HTTP
  - 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

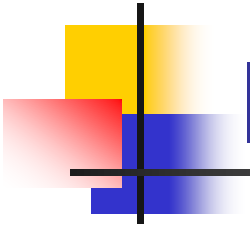
- Maintenant, voyons ce qui se passe lorsque votre page HTML contient du code PHP :
  - Le serveur regarde si le fichier envoyé contient une extension .php
  - Si oui, il transmet le fichier à PHP qui l'analyse et l'exécute
  - Si le code contient des requêtes vers MySQL, PHP envoie la requête SQL à 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

- Intérêt de la base de données
  - La base de données la plus couramment utilisée avec PHP est sans aucun doute MySQL
  - A quoi sert une base de données ?
    - Lorsque vous allez produire des informations dans votre script PHP, vous devez les stocker quelque part
    - Si ce n'est pas le cas, elles seront alors perdues lorsque le serveur renverra la page HTML au client (votre navigateur)
    - Pour les stocker, il existe deux solutions :
      1. les enregistrer dans un fichier texte sur le serveur (quelque part dans l'arborescence de votre hébergement)
      2. les enregistrer dans une base de données
    - La sauvegarde dans un fichier texte n'est pas l'idéal, notamment lorsque vous souhaitez chercher, modifier ou supprimer une partie de l'information que vous stockez

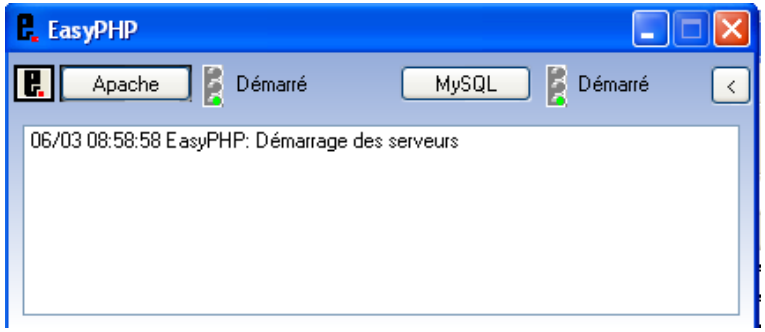


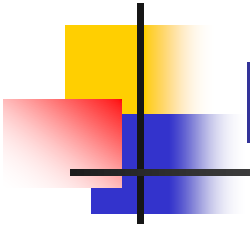
Introduction

- 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

Introduction

- Lancement de EasyPhp
  - Normalement, il est sur le bureau
  - Sinon, aller le chercher dans le répertoire C
  - Cliquer 2 fois sur l'icône, l'interface ci-contre s'affiche
  - Attendre que les feux verts soient allumés
  - L'icône se dessine dans la barre de status en bas à droite de votre écran





Introduction

- Programmes PHP
  - S'installent dans le répertoire WWW de EasyPhp
  - À l'UFR-MI, rangez les programmes sous le répertoire U/EasyPhp/www
  - Pour lancer les programmes, faire clic droit sur E puis Web local
  - L'adresse est :
  - http://127.0.0.1/toto.php
  - Si vous voulez passer un paramètre (param) à toto.php : http://127.0.0.1/toto.php?param=valeur

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

    <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

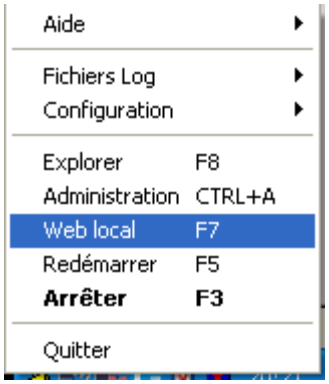
    Sur votre station :

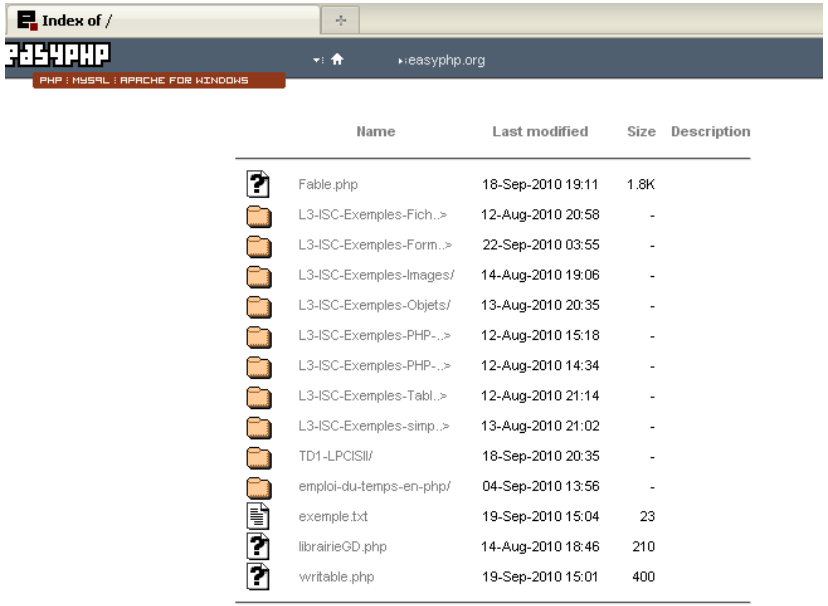
    C:\Program Files\EasyPHP1-8\www\exemple0.php

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



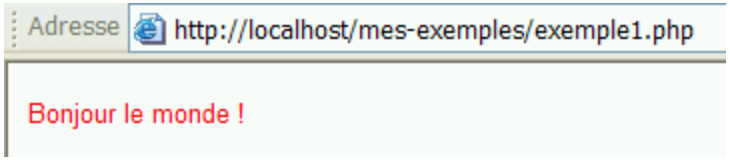


Les bases du PHP

- 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

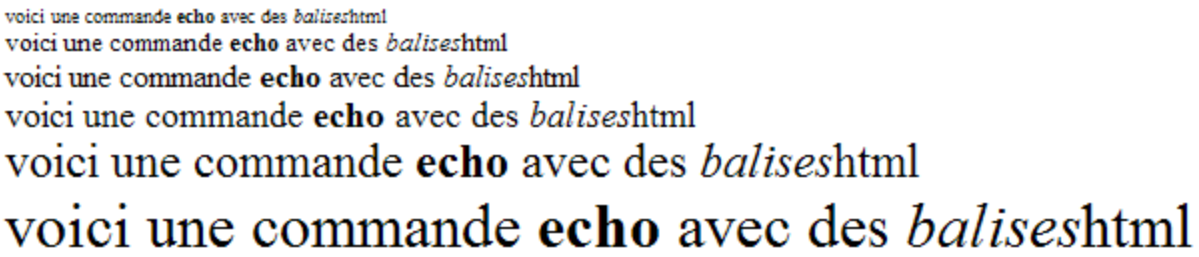
```
<?php
   echo '<div align="center"><font face="arial" size="2"
   color="blue"> Bonjour le monde !</font><br /> ';
   echo '<img src="images/globe.gif" border="0"
   /></div> ';
?>
```

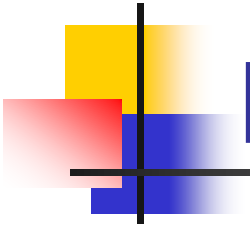
http://www.phpdebutant.org/article14.php

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

- Afficher la date et l'heure
  - Avec PHP, il est facile de manipuler la date
  - Voici un exemple où on voit en même temps comment on manipule les variables
    - Les variables commencent toujours par le signe **$**
    - Par ailleurs, on voit l'usage du Print pour afficher (pareil que echo)

```
<?php
        $date = date("d-m-Y");
        $heure = date("H:i");
        Print("Nous sommes le $date et il est
          $heure");
?>
```

Les bases du PHP

- Constantes et variables
  - Syntaxe des variables

```
<?php
   $variable = "ma première variable" ;
?>
```

  - Déclaration et types
    - PHP n'impose pas de déclarer les variables avant de les utiliser

```
<?php
   $ligne = "droite et courte" ;
   $hauteur = 10;
?>
```

Les bases du PHP

- Portée des variables
  - Variables locales
    - Utilisables uniquement dans les blocs où elles sont déclarées
  - Variables globales
    - Utiles dans les fonctions
    - Exemple : globale.php

```
<?php
    $a = 1;
    function test() {
        global $a;
        $a=2;
        echo $a; /* portée globale */
    }
    test(); //écrit 2 puisque $a est défini en global dans la fonction test()
?>
```

Les bases du PHP

- Portée des variables
  - Autre exemple : globale2.php

```
<?php
   $a = 1;
   $b = 2;
   function somme() {
            global $a;
            $b = $a + $b; //erreur : Undefined variable: b
   }
   somme();
   echo $b; //imprime 2
?>
```

Les bases du PHP

- Constantes et variables
  - Test d'existence
    - La fonction isset() permet de savoir si une variable existe
      - isset($var);
    - Dans le cas où on a affecté une valeur à $var, isset() renvoie TRUE

```
<?php
   $s = "test";
   echo isset($s); //renvoie TRUE
   echo isset($j); //renvoie FALSE
?>
```

Les bases du PHP

- Constantes et variables
  - Destruction
    - La fonction unset() permet de détruire une variable dans le programme
  - Exemple

```
<?php
  $s = "test";
  echo isset($s); //renvoie TRUE
  unset($s);
  echo isset($s); //renvoie FALSE
?>
```

Les bases du PHP

- Constantes et variables
  - Variables dynamiques
    - Sont des variables dont le nom est une variable
  - Exemple : dynamique.php

```
<?php
   $cd = "15 €";
   $dvd = "30 €";
   $produit = "dvd";
   echo $$produit;
?>
```

  - On aurait également pu écrire :

```
echo ${"dvd"};
```

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

- Types de données
  - PHP dispose de quatre types de données simples :
    - booléen, entier, nombre à virgule flottante et chaîne de caractères
    - On peut connaître le type d'une variable à l'aide de : gettype()

```
<?php
   echo gettype("NOM");
   //affiche String
?>
```

Les bases du PHP

- Types de données
  - On peut également utiliser des fonctions d'accès rapide tel que : is_string() qui renvoie TRUE si l'argument est une chaîne de caractères

```
<?php
    $var = 12;
    if(is_string($var)){
   echo "chaine de caractères";
}
else {
    echo "autre type";
}
?>
```

Les bases du PHP

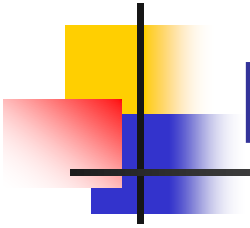
- Interprétation des variables
  - À l'intérieur d'une chaîne entre guillemets, les variables sont automatiquement remplacées par leur valeur
  - Exemple : interpretation.php

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

Les opérateurs

- Opérateurs d'affectation
  - Référence : operateur1.php

```php
<?php
    $origine = 1;
    $reference = & $origine; //marque la référence à la mémoire représentant $origine =1
    $origine = 2; //$reference devient =2
    echo 'Valeur de $reference :', $reference, '<br>';
    //Affiche Valeur de $reference :2
    unset($origine);
    echo 'Valeur de $origine :', $origine, '<br>';
    //Affiche Undefined variable
    echo 'Valeur de $reference :', $reference, '<br>';
    //Valeur de $reference :2
?>
```



Les opérateurs

- Opérateurs arithmétiques
  - Le modulo :
    - renvoie le reste de la division
    - 15%2 : renvoie 1
  - $i++ : équivalent à $i = $i+1
  - $i-- : équivalent à $i = $i-1
  - ++$i : la variable est incrémentée puis évaluée
  - $i++ : la variable est évaluée puis incrémentée
  - $i +=5 : équivalent à : $i = $i+5, de même pour *=, -=, /=

Les opérateurs

- Opérateurs arithmétiques

```
<?php
        $entier   = 7;
        $flottant = 2.5;
        $somme    = 4 + 5; //$somme vaut 9
        $multiplic= 2*5;   //$multiplic vaut 10
        $division = 9/3;   //$division vaut 3
        $modulo   = 10 % 3 //$modulo vaut 1
        $multi_variables = $entier * $flottant; //7 * 2.5 = 17.5
          donc $multi_variables vaut 17.5
?>
```

Les opérateurs

- Opérateurs logiques
  - Exemple | Nom | Résultat
  - $a and $b | ET (And) | Vrai si $a ET $b sont vrai
  - $a or $b | OU (Or) | Vrai si $a OU $b est vrai$
  - a xor $b | XOR (Xor) | Vrai si $a OU $b est vrai, mais pas les deux en même temps
  - ! $a | NON (Not) | Vrai si $a est faux
  - $a && $b | ET (And) | Vrai si $a ET $b sont vrais
  - $a || $b | OU (Or) | Vrai si $a OU $b est vrai
  - La raison pour laquelle il existe deux types de "ET" et de "OU" est qu'ils ont des priorités différentes

Les opérateurs

- Priorité des opérateurs

Priorité des opérateurs								
++++++++++++++++	()	[]						
+++++++++++++++	--	++	!	~	-			
++++++++++++++	*	/	%					
+++++++++++++	+	-						
++++++++++++	<	<=	>=	>				
++++++++++	==	!=						
+++++++++	&							
++++++++	^							
+++++++	|							
++++++	&&							
+++++	||							
++++	?	:						
+++	=	+=	-=	*=	/=	%=	<<=	>
++	AND							
+	XOR							

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 opérateurs de contrôle

== (2 x =)	strictement égale
!=	différent
>	plus grand que
<	inférieur à
>=	supérieur égal à
<=	inférieur ou égal à
&&	et
||	ou
AND	et
OR	ou
TRUE	1 ou oui
FALSE	0 ou non

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

```
<?php
        $homme = FALSE; //booléen ayant la valeur FALS
    E (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 :

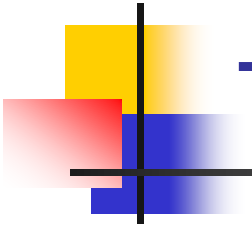
```
<?php
        if($homme === TRUE OR $homme === FALSE &&
    $age > 13) //On veut soit tous les hommes, soit les fille
    s d      e plus de 13 ans
        {
            echo 'Vous pouvez visiter le site';
        }
        else
        {
            echo 'Vous ne pouvez pas visiter le site';
  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

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 l
    es 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.



TD1

- Exercice 1

Structures de contrôle

- Les boucles
  - Première forme

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

  - Deuxième forme

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

  - Troisième forme

```
for
  (expression1;condition;expr
  ession2){
   //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

- Autre exemple avec while()
  - Condition multiple

```
<?php
        $i = 0; //on définit la variable $i qui sera notre n
    ombre que l'on incrémentera. Ici $i va commencer
    à 0
        $j = 4;
        while($i < 7 AND $j < 5)
        {
            echo $i.'<br />';
            $i++;
            $j++;
        }
?>
```

Structures de contrôle

- Exemple avec for()

```
for ($i=0;$i<=4;$i++) // c'est exactement la même chose que dans l'exemple while, mais l'avantage du for est que vous avez tout sur la même ligne, cela évite les oublis d'incrémentation de compteur
{
  print 'boucle numero '.$i.'<br />'; // on affiche une phrase avec le numero de la boucle
}
```

- Affichera à l'écran
  - boucle numero 1
  - boucle numero 2
  - boucle numero 3
  - boucle numero 4

Structures de contrôle

- Autre exemple avec for()

```
<?php
        $i = 0; //on définit la variable $i qui sera notre nombre
                que l'on incrémentera. Ici $i va commencer à 0
        for($j=4; $i < 7 AND $j < 5;$j++)
        {
            echo $i.'<br />';
            $i++;
        }
?>
```

Structures de contrôle

- Exemple avec do while

```
<?php
        $i = 8;
        $j = 7;

        do
        {
            echo 'la boucle a bouclé<br />';
        }
        while($i < $j)
?>
```

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
  - Visibilité des variables
    - Les variables déclarées dans une fonction ne sont utilisables que dans celles-ci
    - Inversement, les variables déclarées dans votre script ne seront pas accessible dans une fonction : les deux espaces sont complètement indépendants
  - Exemple

```
<?php
    $param=3;
    function decremente($valeur){
            $valeur=$valeur-1;
            echo $param; //n'affiche rien
    }
}
decremente($param);
echo $param; //affiche 3
?>
```

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
?>
```

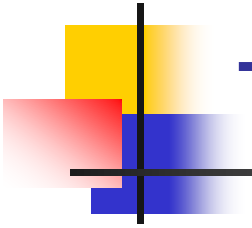
Structures de contrôle

- 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
?>
```



TD1

- Exercice 2-3