Développement serveur en PHP

IO2

Internet et outils

Cristina Sirangelo
IRIF, Université Paris Diderot
cristina@irif.fr

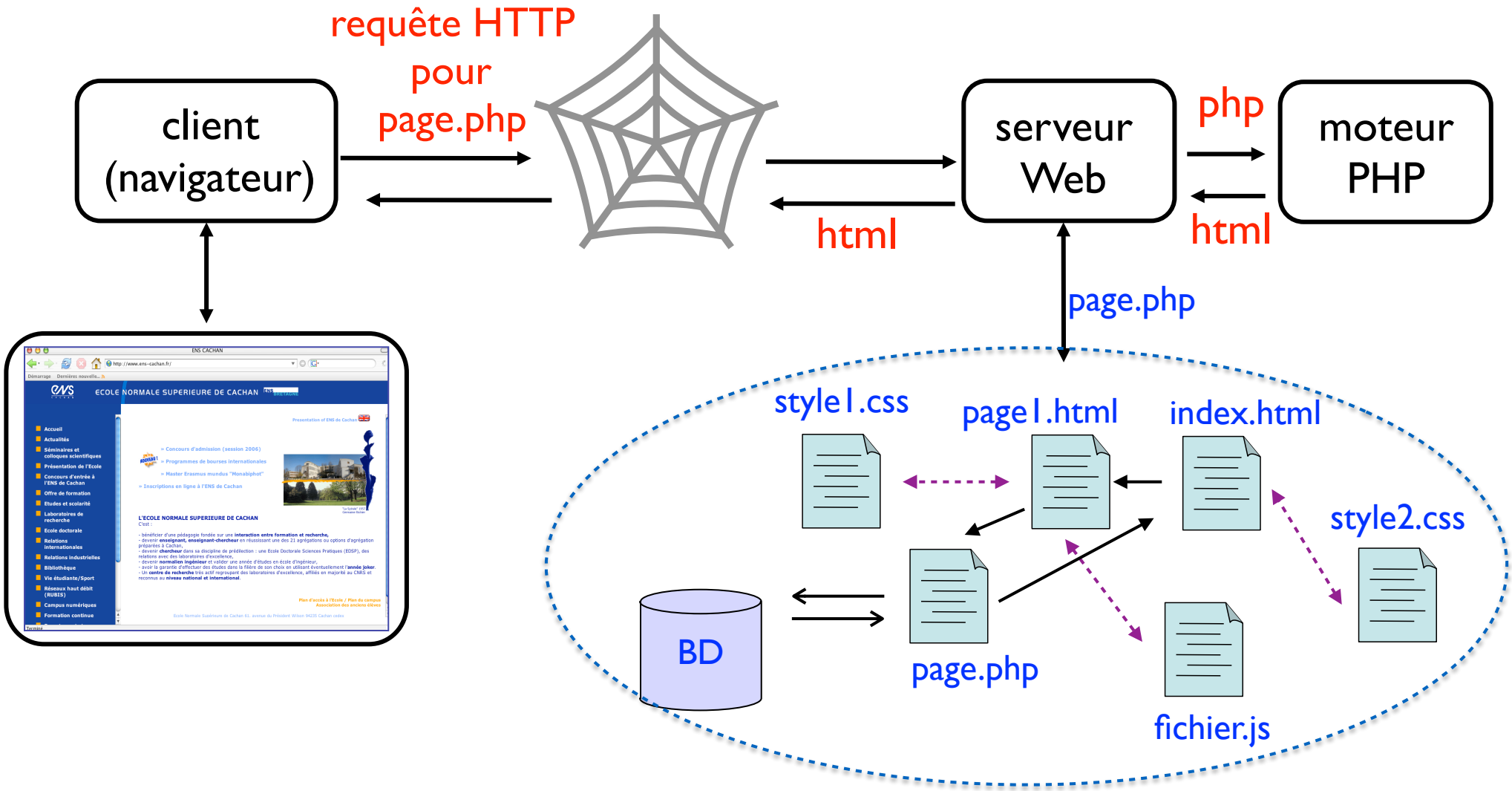
Rappel : structure de base d'une application Web

- Un ensemble de programmes écrits dans un langage de script (e.g. PHP) exécutables par le serveur Web, qui implémentent la logique de l'application et génèrent des documents HTML dynamiques (HTML dynamique coté serveur)
- Eventuellement un ensemble de documents HTML statiques
- HTML statique et dynamique, potentiellement enrichi par
  - des feuilles de style (CSS) pour définir le style des pages
  - des programmes écrits dans un langage (typiquement Javascript) exécutables par le navigateur, qui s'occupent de dynamiser le contenu de la page Web une fois chargée par le navigateur (HTML dynamique coté client)
- Possiblement une base de données pour le stockage des données d'intérêt de l'application
- Tous ces documents et données : accessibles par un serveur Web

PHP est une des langages les plus utilisés pour le développement d'applications Web (du PHP pur au *frameworks*)

Exécution du code coté serveur

- Pour réaliser la logique de l'application, du code est exécuté coté serveur afin de générer les ressources demandées
- Le serveur Web s'appuie sur un moteur de script pour l'interprétation de ce code

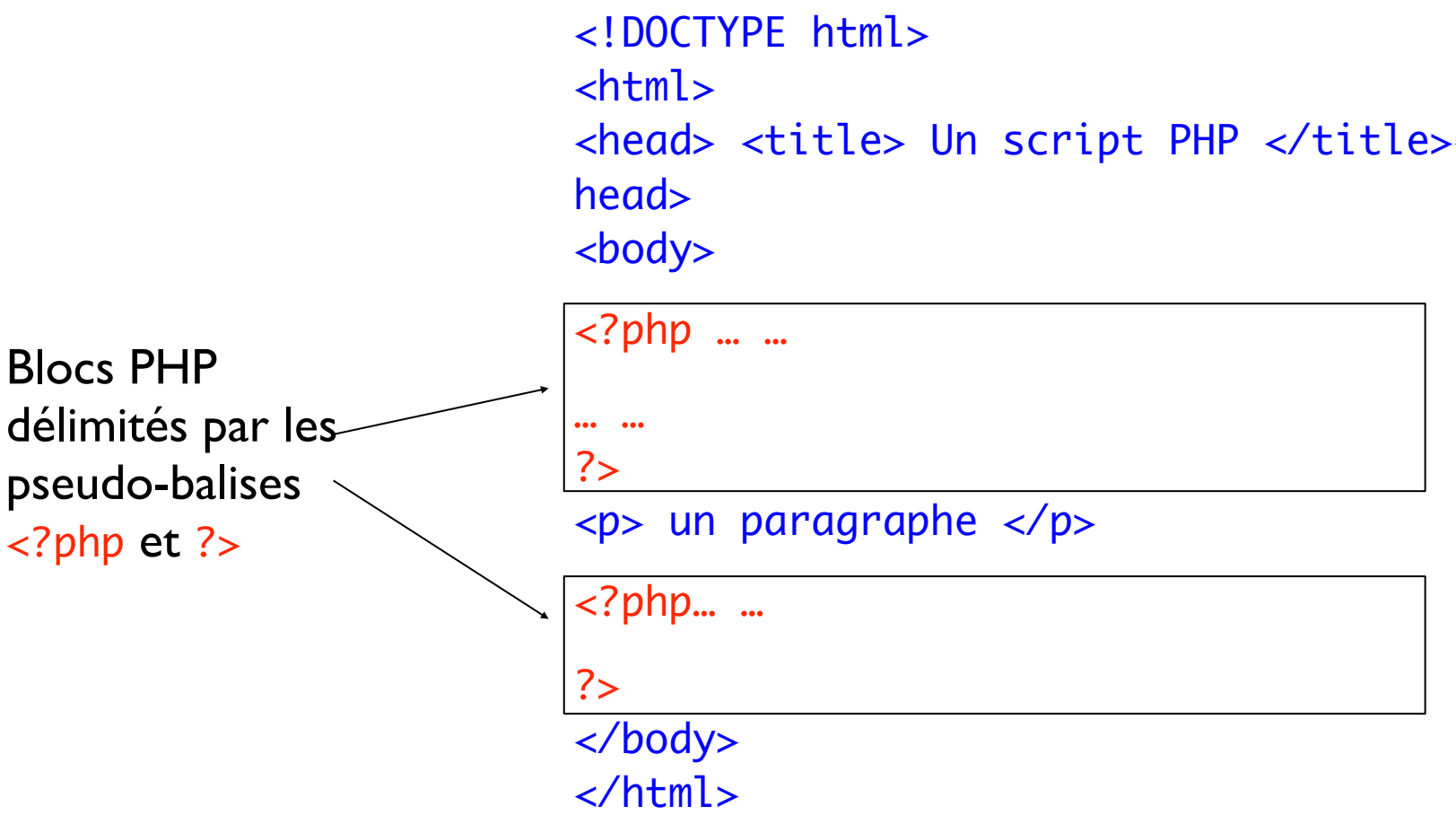


PHP Hypertext Preprocessor

- Langage de programmation à la syntaxe proche du Java
- Langage de script
  - S'utilise et s'exécute côté serveur pour produire un document HTML à renvoyer au client
  - Imbriqué avec le document HTML
  - de nombreux modules d'interface avec d'autres outils dont notamment les serveurs de gestion de bases de données (e.g. MySQL)
- Extension des fichiers : `.php`

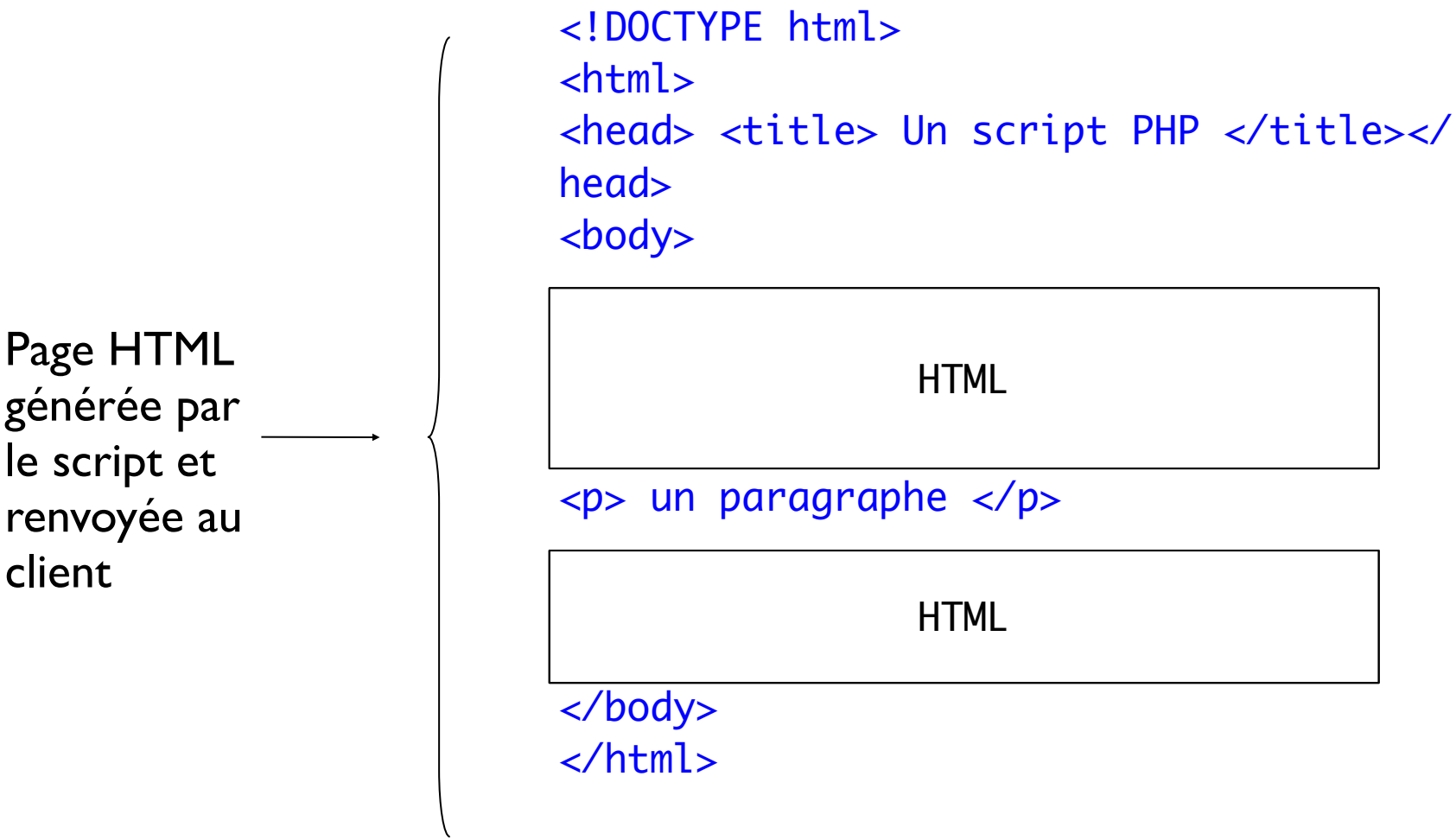
Script PHP

- Un script PHP est un document incluant des fragments de HTML et des blocs d'instructions PHP



Script PHP

- Seulement les blocs PHP sont exécutés et ils renvoient du HTML



Output PHP

- Instruction PHP de base pour produire une chaîne de caractères dans de document :

```
echo (“…”) ;
```

- ou avec plusieurs arguments :

```
echo (“…”, “…”, “…”) ;
```

- parenthèses optionnelles
- La chaîne de caractères peut comporter :
  - des balises HTML
  - des variables PHP qui seront remplacées par leur valeur

Exemple : fichier côté serveur

```
<html>
<head><title>Essai de PHP</title></head>
<body>
<hr>
Nous sommes le <?php echo date ("j / m / Y"); ?>
<hr>
</body>
</html>
```

`date()` est une fonction PHP (prédéfinie)

Exemple : fichier reçu côté client

```
<html>
<head><title>Essai de PHP</title></head>
<body>
<hr>
Nous sommes le  12 / 02 / 2021

<hr>
</body>
</html>
```

04-date.php
localhost/io2/04-date.php

Deuxième exemple

```
<body>
<h1>Un essai de PHP</h1>
<?php
echo "Client : ", $_SERVER["HTTP_USER_AGENT"];
echo "<br><br>Serveur : ";
echo $_SERVER["SERVER_NAME"], "<br>";
?> …
```

- 3 commandes “echo” séparées par des ;
- Arguments de “echo” séparés par des ,
- $_SERVER : une variable PHP prédéfinie (un tableau)

  - contient des information sur le serveur, le client qui demande la page, la requête HTTP, etc.

  - les entrées de ce tableau sont créées par le serveur web et sont disponibles dans tous les scripts (plus de détails plus loin…)

Deuxième exemple (suite)

```
<body>
<h1>Un essai de PHP</h1>
<?php
echo "Client : ", $_SERVER["HTTP_USER_AGENT"];
echo "<br><br>Serveur : ";
echo $_SERVER["SERVER_NAME"], "<br>";
?>
<h3>J'ai réussi mon premier programme PHP
<?php echo "en date du : ", date("d / m / y"), " !" ?>
</h3>
</body>
```

On peut insérer plusieurs séquences PHP dans un fichier

Chaînes entre guillemets : "…"

Code reçu par le navigateur

```
<html>
<head><title>Deuxième exemple de PHP</title></head>

<body>

<h1>Un essai de PHP</h1>

Client : Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_4) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36

<br /><br />Serveur : localhost<br />

<h3>J'ai réussi mon premier programme PHP
 en date du : 12 / 02 / 19 !</h3>

</body></html>
```

En rouge : les parties générées par le code PHP

localhost/io2/04-client-server.php

Rôle du serveur

- Lorsqu'un document HTML est demandé par le client, le serveur se "contente" de trouver le fichier contenant le document correspondant à l'URL et de le renvoyer

- Lorsqu'un script PHP est demandé, le serveur doit d'abord analyser les documents pour :
  - trouver toutes les occurrences de <?php... ?>
  - faire interpréter les commandes PHP par un interpréteur PHP
  - substituer les chaînes <?php... ?> par le résultat de l'interprétation

PHP
Éléments du langage

Aide, manuel PHP

- Le monde PHP est (très) vaste et (très) riche
- Ce cours n'a pour but que de vous donner quelques bases,
- Si vous cherchez une réponse à une question,
- Si vous cherchez une fonction, une syntaxe,
- Consultez :
  - Site officiel : `http://www.php.net/`
  - Pages manuels en français : `https://www.php.net/manual/fr/`
  - `http://fr.wikipedia.org/wiki/PHP`

Variables

- Noms de variables :
  - Commencent par `$`
  - Longueur quelconque
  - Composés de lettres, de chiffres, et `_`
  - Lettre ou `_` après le `$`
  - Sensibles à la casse (majuscules et minuscules ne sont pas identiques)

- Exemples
  - `$ma_var, $Ma_Var, $mon_1er_nom, $_une_autre`

Variables

- Créer des variables (pas de déclaration de type!) :

```
$note = 19;
$coeff = 1.53;
$nom = "Roger Rabbit";
echo($note);
echo " Note : ", $note, " Nom : ", $nom,
     " c : ", $koeff;
```

- Une variable pas définie (`$koeff`) n'a pas de valeur (pas affichée par `echo`)
- Selon les réglages de l'interpréter PHP, l'utilisation d'une variable non-définie peut générer un message d'alerte :

Notice: Undefined variable: koeff in xxx.php on line 12

localhost/io2/04-variables.php

Variables

- Supprimer des variables:
  - `unset($note);`
  - `unset($note, $coeff, $nom);`

- En PHP une variable peut être :
  - définie avec une valeur : `$var = "toto";`
  - définie et vide : `$var1 = “”; $var2 = 0; $var3 = false; …`
  - non définie : `$var;` *variable sans valeur*

- Vérifier le contenu d'une variable :

```
isset($var)  // Vrai si définie, même vide
empty($var)  // Vrai si vide, ou pas définie
```

Variables : Exemple

```php
$note = 19;
$coeff;
$nom ="";
echo
isset($note), // vrai
isset($nom), //vrai
isset($newvar), //faux
 isset($coeff), //faux
empty($coeff), // vrai
empty($nom), //vrai
empty($newvar); //vrai
```

Types de données

- Il existe plusieurs types de base:
  - Entier (integer) : `$note = 1728;`
  - Nombre à virgule flottante (float ou double) : `$temp = 37.6;`
  - Chaîne de caractères (string) : `$nom = "personne";`
  - Booléen (boolean) : `$condition = TRUE; $condition = FALSE;`
- ainsi que des types composés : Tableaux (`array`), Objet (`object`), …

- Pas de déclaration de type : le type d'une variable est déterminé par son "contenu"
- La même variable peut prendre des types différents au cours de son cycle de vie :

```
$note = 17;    //$note est de type entier
$note = "très bien"; //$note est maintenant de type string
```

Vérifier le type d'une variable

```
is_int($var)  // Vrai si type entier
is_float($var)
is_string($str)
…
```

`gettype($var)` renvoie le type de la variable `$var`

Conversion de type

```
$entier = (integer)$var2;

$chaine = (string)$var3;

$flottant = (float)$var4;
```

Attention, on ne peut pas convertir n'importe quelle chaîne en entier (0 si impossible)

```
$entier = (int)"1597"; // OK, $entier = 1597

$entier = (int)"mille deux cents"; // non! $entier = 0

$entier = (int)"1597 mille deux cents"; // OK!
```

Exemple

```php
<?php

    $entier1 = 4; $entier2 = 44;
    $chaine = "2021 une bonne année!";
    $string = "une bonne année 2021”;

    $entier1 = $chaine; $entier2 = $string;

    echo "entier1 : ", $entier1, "<br />”,
         "entier2 : ", $entier2, "<br />";

    $entier1 = (int)$chaine; $entier2 = (int)$string;

    echo "entier1 : ", $entier1, "<br />”,
         "entier2 : ", $entier2; "<br />”;

?>
```

localhost/…/04-casting.php

Constantes

```
define("nom_cste", valeur_cste);
```

Ne doivent pas correspondre aux mots-clés de PHP

Exemple :

```
define("USD_TO_EUR", 0.929);
echo "5 dollars valent : ",  5 * USD_TO_EUR,
     "euros";
```

⇒ 5 dollars valent : 4.64 euros

Commentaires

- Insérer des commentaires en PHP:

```
/* ceci est un
commentaire */
// commentaire jusqu'à la fin de la ligne
# Ceci également
```

Calculs arithmétiques

Opérateurs :

```
+ - * / %
$cpt++; // équivalent à $cpt = $cpt + 1;
$cpt--; // équivalent à $cpt = $cpt - 1;
$cpt /= 2; $cpt += 77; $cpt -= 2; $cpt *= 5;
```

Quelques fonctions :

```
sqrt(64) // => 8 racine carrée
ceil(27.68) // => 28 entier immédiatement supérieur
floor(27.9) // => 27 entier immédiatement inférieur
round(24.62) // => 25
```

Opérateurs et types

- Attention : certains opérateurs peuvent changer automatiquement le type d'une variable :

```
$cpt = 7; echo gettype($cpt), "<br>";
$cpt /= 2; echo gettype ($cpt), "<br>";
$cpt = 8; echo gettype($cpt), "<br>";
$cpt /= 2; echo gettype ($cpt), "<br>";
$str = "10";echo gettype ($str), "<br>";
$str += 5; echo gettype ($str), “<br>";
```

- Remarque : `/` sur un entier n'est pas la division entière !

localhost/…/04-op.php

Chaînes de caractères

- Deux formes : '…' et "…"
- Différence : "…" admet interpolation des variables

Chaînes de caractères

- Deux formes : '…' et "…"
- Différence : "…" admet interpolation des variables
  - une variable présente dans "…" sera remplacée par sa valeur

Chaînes de caractères

- Deux formes : '…' et "…"
- Différence : "…" admet interpolation des variables
  - une variable présente dans "…" sera remplacée par sa valeur

```
$var = "jean"
```

Chaînes de caractères

- Deux formes : '…' et "…"
- Différence : "…" admet interpolation des variables
  - une variable présente dans "…" sera remplacée par sa valeur

```
$var = "jean"
echo "Mon nom est $var <br>"
```

Chaînes de caractères

- Deux formes : '…' et "…"
- Différence : "…" admet interpolation des variables
  - une variable présente dans "…" sera remplacée par sa valeur

```
$var = "jean"
echo "Mon nom est $var <br>"
```

⇒ Mon nom est jean

Chaînes de caractères

- Deux formes : '…' et "…"
- Différence : "…" admet interpolation des variables
  - une variable présente dans "…" sera remplacée par sa valeur

```
$var = "jean"
echo "Mon nom est $var <br>"
```

⇒ Mon nom est jean

```
echo 'Mon nom est $var <br>'
```

Chaînes de caractères

- Deux formes : '…' et "…"
- Différence : "…" admet interpolation des variables
  - une variable présente dans "…" sera remplacée par sa valeur

```
$var = "jean"
echo "Mon nom est $var <br>"
```

⇒ Mon nom est jean

```
echo 'Mon nom est $var <br>'
```

⇒ Mon nom est $var

Chaînes de caractères

- Deux formes : '…' et "…"
- Différence : "…" admet interpolation des variables
  - une variable présente dans "…" sera remplacée par sa valeur

```
$var = "jean"
echo "Mon nom est $var <br>"
```

⇒ Mon nom est jean

```
echo 'Mon nom est $var <br>'
```

⇒ Mon nom est $var

- Les guillemets doubles doivent être échappées dans "…" mais pas dans '…'
- Les guillemets simples doivent être échappées dans '…' mais pas dans "…"

```
$str = "Cette chaîne n'a qu'un \" guillemet double";
$str = 'Cette chaîne "s\'écrit" avec une apostrophe';
```

Chaînes de caractères

- Concaténer des chaînes :

```
$str1 = "Hello";
$str2 = "World!";
$salut = $str1 ." " . $str2;
echo $salut;
```

⇒ Hello World!

- Taille d'une chaîne de caractères (nombre de caractères): `strlen()`

```
echo strlen("Hello world!");
```

⇒ 12

- php offre plusieurs fonctions natives pour travailler avec les chaînes de caractères (cf. plus loin)

Tableaux (array)

- Stockent plusieurs valeurs dans une seule variable
- Les tableaux php peuvent stocker des valeurs de différents types
- Deux formes :
  - indicés : valeurs associés à des indices 0, 1, 2…
  - associatifs : valeurs associes à des “clefs” (chaînes de caractères), comme une table de hachage
- Si `$t` est un tableau, `count($t)` renvoie le nombre d’éléments de `$t`
- Comme toute autre variable en PHP, un tableau est crée par affectation de son contenu

Tableaux : création

Créer un tableau indicé :

```
$films[0] = "Casablanca";
$films[1] = "To Have and Have Not";
```

Pas d'obligation de commencer à 0

Créer un tableau associatif :

```
$capitale["FR"] = "Paris";
$capitale["UK"] = "Londres";
$capitale["IT"] = “Rome";
```

Indices et clefs peuvent coexister dans le même tableau :

```
$rues[4] = "Rue Lecourbe”
$rues[“4bis”] = "Rue de la Paix”
```

Tableaux : création

Créer sans indice explicite :

```
$livres[] = "Les Misérables";
$livres[] = "Notre Dame de Paris";
$livres[] = "Quatre-vingt treize";


echo $livres[0] ⇒ Les Misérables
echo $livres[1] ⇒ Notre Dame de Paris
echo $livres[2] ⇒ Quatre-vingt treize
```

Tableaux : création

Avec le constructeur `array` (en une seule fois) :

- Tableau indicé :

```
$livres = array("Les Misérables",
                "Notre Dame de Paris",
                "Quatre-vingt treize");

echo $livres[0] ⇒ Les Misérables
```

- Tableau associatif :

```
$capitales = array("FR"=> "Paris", "UK"=> “Londre”, "IT"=> "Rome");
```

- Tableau mixte :

```
$rues = array(4 => "Rue Lecourbe",  "4bis" => "Rue de la Paix");
```

- Indices implicites :

```
$rues = array(4 => "Rue Lecourbe", "Rue de la Paix");
echo $rues[5] ⇒ Rue de la Paix

$rues = array(“4bis” => "Rue de la Paix", “Rue Lecourbe");
echo $rues[0] ⇒ Rue de la Paix
```

Tableaux : création

Intervalles :

```
$annees = range(1970, 2038);
```

On peut aussi créer des tableaux multi-dimensionnels (tableaux de tableaux)

```
$personnes = array
  (
   array("nom"=>"Dupont", "prenom" => "Jean"),
   array("nom"=>"Portier", "prenom" => "Louis", "age" => 39)
  );

echo $personnes[1]["nom"]
```

⇒ Portier

Tableaux : création

Intervalles :

```
$annees = range(1970, 2038);
```

On peut aussi créer des tableaux multi-dimensionnels (tableaux de tableaux)

```
$notes = array
  (
   “Dupont” => array(“Exo1” => 12, “Exo2” => 14),
   “Portier” => array(“Exo1”=>16, “Exo2” => 18, “Exo3” => 15)
  );

echo $notes[“Portier”][“Exo3”];
```

⇒ 15

Tableaux : création

Création de tableaux multi-dimensionnels par affectation (même effet):

```
$personne[0]["nom"] = "Jean";
$personne[0]["prenom"] = "Dupont";
$personne[1]["nom"] = "Louis";
$personne[1]["prenom"] = "Portier";
$personne[1]["age"] = 39;
```

```
$notes["Dupont"]["Exo1"] = 12; $notes["Dupont"]["Exo2"] = 14;
$notes["Portier"]["Exo1'] = 16; $notes["Portier"]["Exo2"]= 18;
$notes["Portier"]["Exo3"] = 15;
```

Tableaux : impression

```
print_r($livres)
```

```
⇒ Array ( [0] => Les Misérables
          [1] => Notre Dame de Paris
          [2] => Quatre-vingt treize )
```

```
var_dump($livres)
```

```
⇒ array(3) { [0]=>  string(14) "Les Misérables"
             [1]=>  string(19) "Notre Dame de Paris"
             [2]=>  string(19) "Quatre-vingt treize" }
```

Tableaux : modification

- Modification d'un valeur :

```
$livres[1] = "";

print_r($livres);

⇒ Array ( [0] => Les Misérables
          [1] =>
          [2] => Quatre-vingt treize )
```

- Suppression d'une case :

```
unset($livres[1]);

print_r($livres);

⇒ Array ( [0] => Les Misérables
          [2] => Quatre-vingt treize )
```

Tableaux : parcours

Une variante de la boucle `for` permet de parcourir aisément les tableaux :

```
foreach($array as $var) {
     instructions
}
```

- une itération pour chaque case du tableau
- à l'itération i, `$var` prend la valeur de la i-eme case du tableau `$array`
- s'applique aussi bien aux tableaux indicés qu'aux tableaux associatifs

Tableaux : parcours

- Exemple avec tableau indicé :

```
$livres = array("Les Misérables",
                "Notre Dame de Paris",
                "Quatre-vingt treize");

foreach($livres as $titre) {
  echo $titre."  ";
};
```

⇒ Les Misérables Notre Dame de Paris Quatre-vingt treize

- Exemple avec tableau associatif :

```
$capitales = array('FR'=> "Paris", 'UK'=> “Londre”, 'IT'=> "Rome");

foreach($capitales as $ville) {
  echo $ville." ";
}
```

⇒ Paris Londres Rome

Tableaux : parcours

Pour récupérer à chaque itération à la fois la valeur et l'indice/clef de la case :

```
foreach($array as $index => $var) {
    instructions
}
```

- Exemple avec tableau indicé :

```
$livres = array("Les Misérables",
                "Notre Dame de Paris",
                "Quatre-vingt treize");
foreach($livre as $index => $titre) {
   echo "Livre #", $index, " : $titre<br/>";
}
```

```
⇒ Livre #0 : Les Misérables
  Livre #1 : Notre Dame de Paris
  Livre #2 : Quatre-vingt treize
```

Tableaux : parcours

Pour récupérer à chaque itération à la fois la valeur et l'indice/clef de la case :

```
foreach($array as $index => $var) {
      instructions
}
```

- Exemple avec tableau associatif :

```
$capitales = array('FR'=> "Paris", 'UK'=> “Londre”, 'IT'=> “Rome");
foreach($capitales as $pays => $ville) {
 echo "capitale de ", $pays, " : $ville<br/>";
}
```

```
⇒ capitale de FR : Paris
  capitale de UK: Londres
  capitale de IT : Rome
```

Tableaux : parcours

Exercice.

Donné un tableau contenant un nombre arbitraire de lignes de la forme :

```
$notes = array
  (
   “Dupont” => array(“Exo1” => 12, “Exo2” => 14),
   “Portier” => array(“Exo1”=>16, “Exo2” => 18, “Exo3” => 15)
 );
```

Calculer un tableau associatif contenant pour chaque étudiant, la moyenne de ses notes ;

l’afficher d’abord avec `print_r()`; ensuite comme un tableau HTML

04-moyenne.php

Tableaux : parcours

Par manipulation d'un pointeur interne au tableau :

```
current($array); // renvoie la valeur sous le pointeur

next($array); previous($array);

   //avance/recule le pointeur et renvoie la valeur sous le pointeur
   après le déplacement. Renvoie false s'il n'est pas possible
   d'avancer/reculer

reset($array); end($array);

  //positionne le pointeur sur le premier/dernier élément du tableau
```

Tableaux : parcours

Par manipulation d'un pointeur interne au tableau :

```
current($array); // renvoie la valeur sous le pointeur

next($array); previous($array);

   //avance/recule le pointeur et renvoie la valeur sous le pointeur
   après le déplacement. Renvoie false s'il n'est pas possible
   d'avancer/reculer

reset($array); end($array);

  //positionne le pointeur sur le premier/dernier élément du tableau
```

Exemple :

```
$livres = array("Les Misérables", "Notre Dame de Paris",
                                          "Quatre-vingt treize");

reset($livres);

echo current($livres);  // ⇒ Les Misérables

echo next($livres);  // ⇒ Notre Dame de Paris

echo next($livres);  // ⇒ Quatre-vingt treize
```

Tableaux prédéfinis (superglobals)

Tout script PHP a accès à des tableaux associatifs prédéfinis, rendus disponibles par le serveur Web, dont voici quelques uns :

`$_SERVER`

Informations relatives au serveur et à la connexion

Exemple :

```
$_SERVER["HTTP_USER_AGENT"]; $_SERVER["SERVER_NAME"]
$_SERVER["SCRIPT_NAME"] //le chemin du script courant
```

Tableaux prédéfinis (superglobals)

Tout script PHP a accès à des tableaux associatifs prédéfinis, rendus disponibles par le serveur Web, dont voici quelques uns :

`$_GET`

les paramètres passées au script dans l'url :

```
http://host/chemin/page.php?clef1=val1&clef2=val2
```

Dans page.php :

```
$_GET['clef1']  // a valeur 'val1'
$_GET['clef2']  // a valeur 'val2'
```

Tableaux prédéfinis : exemple 1

Affiche “Bonjour nom prénom !”

```
<body>
<h2> Bonjour

<?php
echo $_GET['prenom’].’ '. $_GET['nom'].' !’;
?>

</h2>
</body>
```

http://localhost/.../04-get.php?prenom=Jean&nom=Dupont

Tableaux prédéfinis : exemple 2

```
<head>
<style>
body{
background-color:
<?php
echo $_GET['col'];
?>
}
</style>
</head>

<body>
<h2> Bonjour
<?php
echo $_GET['prenom'].' '. $_GET['nom'].' !';
?>
</h2>
</body>
```

Affiche Bonjour nom prénom !
avec une couleur de fond passée comme paramètre

http://localhost/.../04-getcol.php?prenom=Jean&nom=Dupont&col=yellow

Tableaux prédéfinis (superglobals)

Tout script PHP a accès à des tableaux associatifs prédéfinis, rendus disponibles par le serveur Web, dont voici quelques uns :

`$_SERVER, $_GET,`

…et d'autres dont on parlera plus tard

`$_POST, $_COOKIE, $_REQUEST, $_SESSION, …`

Contrôle du flux d'exécution : instruction conditionnelle

```
if (condition) {
   // code exécuté si la condition est vraie
} else {
   // code exécuté si la condition est fausse
}
```

Contrôle du flux d'exécution : instruction conditionnelle

Conditions :

Égalité de valeur : `$v1 == $v2`

Égalité de valeur **et** de type : `$v1 === $v2`

Différence de valeur : `$v1 != $v2`

Différence de valeur **ou** de type : `$v1 !== $v2`

Comparaisons : `>` , `<` , `>=`, `<=`

Opérateurs logiques : `&&` , `||`, `!`

Contrôle du flux d'exécution : instruction conditionnelle

Exemple d'utilité de ===

- Certaines fonctions peuvent renvoyer soit une valeur, soit FALSE si l'opération ne réussit pas.
- === nous permet de distinguer entre FALSE et une valeur équivalente à FALSE

Exemple : la fonction `next($tab)`

- la condition `next($tab) == false` nous permet de comprendre s'il y a un prochain élément dans le tableau ou pas
- cependant cette condition effectue d'abord une conversion de type : `next($tab)` est d'abord converti en boolean

Qu'est-ce qui se passe si le tableau contient une valeur 0 par exemple?

=== nous permet de résoudre ce problème

04-cond.php
http://localhost/.../04-cond.php

Contrôle du flux d'exécution : instruction conditionnelle

```
<body>
<h2>
<?php
if ($_GET['user'] == 'jean') {
    echo "Bienvenue !";
} else {
    echo "Acces interdit";
}
?>
</h2>
</body>
```

Affiche Bienvenue !
à un seul utilisateur

Remarque : ce n'est pas une bonne idée de passer des informations sensibles (pwd, …) comme paramètre dans l'url (visible par tous), cf. plus loin

http://localhost/.../04-if.php?user=jean

Mettre “en pause” l’interpréteur PHP

On a déjà vu qu’un script php peut alterner des fragments de code et du HTML :

```
<body>
<h1>Un essai de PHP</h1>
<p> Bonjour
<?php
 echo $_GET[‘prenom'].' '. $_GET['nom'].' !';
?>
</p>
<h2>Vous vous êtes connectés en date du :
<?php echo date("d / m / y"), " !" ?>
</h2>
</body>
```

Ce qui est pratique pour éviter des longues chaines de caractères dans les echo

Mettre “en pause” l’interpréteur PHP

Script équivalent au précédent mais moins lisible :

```
<body>
<h1>Un essai de PHP</h1>
<p> Bonjour
<?php
 echo $_GET[‘prenom'].' '. $_GET[‘nom'].' !’. ‘</p> <h2>Vous vous êtes
connectés en date du :’;
echo date("d / m / y"), " !" ;
?>
</h2>
</body>
```

Mettre “en pause” l’interpréteur PHP

L’interprète PHP peut être mis en pause également au milieu d’une instruction de contrôle du flux d’exécution (if, while etc.) :

Reprenons l’exemple de l’affichage à un seul utilisateur :

```
<body>
<?php
if ($_GET[‘user’] == ‘jean’) {
   echo “<h2>Bienvenue !</h2>”;
} else {
   echo “<h2>Acces interdit</h2>“;
}
?>
</body>
```

Supposons que dans le premier cas on veut afficher pas uniquement “Bienvenue !” mais du contenu HTML plus complexe (e.g. une “grande” table)

Mettre “en pause” l’interpréteur PHP

```
<body>
<?php
if ($_GET[‘user’] == ‘jean’) {
   echo “<h2>Bienvenue !</h2>”;
?>

<table border="1">
<caption>Une table</caption>
<tr><th colspan="2">Lorem ipsum</th><th>dolor</th></tr>
<tr><td>sit</td><td>amet</td><td rowspan="2">consectetur Cras</td></
tr>
<tr><td>adipiscing</td><td>elit</td></tr>
<tr><td>ultrices</td><td>lacus</td><td>...</td></tr>
</table>

<?php
} else {
   echo “<h2>Acces interdit</h2>“;
}
?>
```

http://localhost/.../04-pause.php?user=jean

Contrôle du flux d'exécution : conditionnelle `switch`

```
switch ($var) {
  case valeur1:
    /* bloc… */
    break;
  case valeur2:
    /* bloc… */
    break;
  default:
    /* bloc… */
}
```

- La valeur de `$var` est comparée avec `valeur1`, `valeur2`, …
- quand il y a égalité, le bloc d'instructions correspondant est exécuté
- `break;` est nécessaire pour éviter que le bloc suivant soit également exécuté
- le bloc `default` est exécuté si il n'y a pas d'égalité

switch : exemple

```
<head><style>
body{
   background-color:
   <?php
   switch ($_GET['col']) {
      case '0' :
         echo "red";
         break;
      case '1':
         echo "yellow";
         break;
      case '2':
         echo "lightblue";
         break;
      default :
         echo "pink";
   }
   ?>
}
</style></head> …
```

Affiche une couleur de fond différente selon la valeur d'un code passé en paramètre

http://localhost/.../04-switch.php?col=0

Boucles

```
for (initialisation; condition; fin d'itération) {
  /* bloc… */
}

while (condition) {
  /* bloc… */
}

do {
  /* bloc… */
} while (condition);
```

Contrôle de boucle :

```
continue; // passe à l'itération suivante

break; // sort de la boucle
```

http://localhost/.../04-boucles.php?n=6

04-boucles.php

PHP
fonctions et fonctions natives

Fonctions

Une façon de nommer un calcul

Le code peut ensuite être écrit en faisant abstraction de comment ce calcul réalisé

Avantages :

- Rendre le code modulaire
- Eviter de réécrire du code identique (factorisation de code)
- Eviter des erreurs
- Rendre le code plus lisible
- Gagner du temps

Définition / Utilisation de fonction

Définition d'une fonction

```
function nom_fonction($arg1, $arg2, …) {
  /* bloc… */
}
```

Appel (utilisation) d'une fonction

```
nom_fonction($valeur1, $valeur2, …);
```

Exemple

```
<? php
  function milieu($a, $b) {
    return floor( ($a+$b) / 2 );
  }
  $x= 31; $y = 50;  echo milieu ($x, $y);
?>
```

⇒ 40

Fonctions et méthodologie de développement

- En PHP les fonctions sont particulièrement utiles pour aider à séparer la logique de l'application (le code PHP) de la présentation (le HTML)
- Une bonne méthodologie de développement :
  - Définir toujours une bibliothèque de fonctions effectuant les différentes taches nécessaires (affichage du HTML, connexion à la bases de données, calculs spécifique, etc.)
  - ensuite appeler ces fonctions pour générer la page demandée selon la logique des l'application

04-fonctions.php

- Cette méthodologie peut être poussée jusqu'à définir des structures très génériques de serveurs (pseudo-frames, MVC, …) qui sont aujourd'hui à la base de la plupart des *frameworks* pour le développement Web

Passage des paramètres à une fonction

Par défaut : par valeur (y compris pour les tableaux !)

- La fonction modifie une copie locale du paramètre, cela n'affecte pas la valeur du paramètre en dehors de la fonction

```
function modifie ($tableau) {
     $tableau[0]= 2;
}

$t= array(0,1);

modifie($t);

foreach($t as $case) {
     echo $case," ";
}
```

⇒ 0 1

Passage des paramètres à une fonction : par référence

On peut passer à une fonction une référence vers la variable à mettre à jour

- La fonction modifie la variable elle-même, et non pas une copie locale, cela affecte la valeur du paramètre en dehors de la fonction

```
function modifie (&$tableau) {
     $tableau[0]= 2;
}

$t= array(0,1);

modifie($t);

foreach($t as $case) {
     echo $case," ";
}
```

⇒ 2 1

- Passage par référence possible pour n'importe quelle variable, mais pas pour une constante

Visibilité des variables

```
$a = 6789; //variable globale

function f ($arg1, $arg2) {

  $b = 12345; // variable locale

  echo $a; // n'affiche rien, $a ici est une nouvelle variable locale
           // pas définie

     …

}

echo $b; //n'affiche rien, $b ici est une nouvelle variable globale
       // pas définie
```

- Les variables internes à une fonction ne sont pas visibles de l'extérieur de la fonction
- Les variables externes ne sont pas visibles dans la fonction
- Une fonction peut acquérir la visibilité des variables globales avec le mot clef "`global`"

Visibilité des variables

```
$a = 6789; //variable globale
function f ($arg1, $arg2) {
  global $a; //variable globale $a rendue visible dans f()
  $a = 33;
      …
}
f(2, 3);
echo $a;
```

⇒ 33

Modules PHP et extensions

- Un module est une bibliothèque de fonctions disponibles dans le langage
- L'installation standard de PHP (le "core") dispose de plusieurs modules natifs (e.g. des fonctions pour travailler avec les chaînes de caractères, les tableaux, …)
- Il est également possible d'installer un ensemble très riche de modules d'extension
- Voici quelques fonctions utiles appartenant au modules natifs…

Fonction date

`date($format, $timestamp)`

`$format` : une chaîne de la forme `"spec[-/.]spec[-/.]…"`

Les spécificateurs sont séparés par `-` ou `/` ou `.`

- renvoie la date `$timestamp` sous forme d'une chaîne, au format donné par `$format`
- `$timestamp` est un entier
  - nombre de secondes depuis 1er jan. 1970 00:00:00 GMT
  - optionnel
    - valeur per défaut : la date courante

Fonction date

- Spécificateurs : (liste non exhaustive)

M	abréviation mois en anglais
F	mois en toute lettre en anglais
d	jour dans le mois
l	jour de la semaine en lettres en anglais
Y	année sur 4 chiffres
H	heures entre 0 et 24
h	heures entre 0 et 12
i	minutes
s	secondes

- Exemple

```
date('d / M / Y');  // => 14 / Feb / 2020
date('l / F-d-Y / H.i'); // => Friday / February-14-2020 / 11.15
```

Quelques fonctions natives sur les chaînes de caractères

- `strlen($str)`

  Renvoie la longueur de la chaîne `$str`

  `strlen("ab de  ")` ⇒ 7

- `substr($str, $debut [,$long])`

  Renvoie le segment de la chaîne `$str` qui commence à l'indice `$debut` et comprend `$long` caractères. Les indices démarrent à 0.

  `substr("abcdefgh", 2, 3)` ⇒ cde

- `strpos($chaine1, $chaine2)`

  Renvoie :

  - l'indice du premier caractère de `$chaine2` dans `$chaine1`, si `$chaine2` est contenue dans `$chaine1`

  - `FALSE` sinon

  Exemple : `strpos("aaabbbccddeebbb", "bb")` => 3

Quelques fonctions natives sur les chaînes de caractères

- `strchr($bottedefoin, $aiguille)` ou
- `strstr($bottedefoin, $aiguille)` (synonymes)
  - Recherche la chaîne `$aiguille` dans celle `$bottedefoin`
  - Renvoie :
    - la sous-chaîne de `$bottedefoin` commençant à la première occurrence de `$aiguille`
    - ou `FALSE` si `$aiguille` n'est pas trouvée
  - Exemple `strchr("www.univ-paris-diderot.fr", ".")`

    ⇒ `.univ-paris-diderot.fr`
- `strrchr($tasdefoin, $aiguille)`

  Idem, mais travaille sur la dernière occurrence

  `strrchr("www.univ-paris-diderot.fr", ".");`

  ⇒ `.fr`

Quelques fonctions natives sur les chaînes de caractères

- `strcmp($str1, $str2)`

  Renvoie 0 si les deux chaînes sont égales
  une valeur > 0 si `$str1` > `$str2`
  une valeur < 0 si `$str1` < `$str2`

  `strcmp("abcde", "bcdef")` ⇒ `-1`

  `strcmp("abcde", "Abcde")` ⇒ `1`

  Remarque. Pour tester uniquement l'égalité de chaînes de caractères :

  == ou === (équivalents si les deux arguments sont du même type)

Quelques fonctions natives sur les chaînes de caractères

- `str_repeat($str, $num)`

  `str_repeat("xyz", 5)` ⇒ xyzxyzxyzxyzxyz

- `str_replace($a, $b, $str)`

  Remplace toutes les occurrences de la chaîne `$a` par la chaîne `$b` dans la chaîne `$str`

  `$a` et/ou `$b` peuvent être des tableaux

```
str_replace("m", "M", "Lorem Ipsum")  ⇒ LoreM IpsuM

str_replace(array("ab", "d"), array("c","f"), “abd");
    ⇒ cf

str_replace(array("ab", "d"), "c", “abd");
    ⇒ cc
```

Quelques fonctions natives sur les chaînes de caractères

- Supprimer des caractères “invisibles” (espaces) :

```
$str_out = trim($chaine) // supprime au début et à la fin de la chaîne
$str_out = ltrim($chaine) // supprime uniquement au début (gauche)
$str_out = rtrim($chaine) // supprime uniquement à la fin (droite)
```

- Changer la casse

`strtolower($str)`

Renvoie la chaîne `$str` après avoir converti les majuscules en minuscules

```
strtolower("LorEM IPsum, dOlor sit aMEt")
```

⇒ lorem ipsum, dolor sit amet

`strtoupper($str)`

Renvoie la chaîne `$str` après avoir converti les minuscules en majuscules

```
strtolower("LorEM IPsum, dOlor sit aMEt")
```

⇒ LOREM IPSUM, DOLOR SIT AMET

Quelques fonctions natives sur les chaînes de caractères

Manipulation de chaînes pour HTML :

`htmlspecialchars($str)`

- Renvoie une chaîne où <, >, &, et " sont remplacés par des entités HTML (`<` `>` `&` et `&quot;`)

- Avec l'option `htmlspecialchars($str, ENT_QUOTES)`, les guillemets simples ' sont également remplacés par l'entité correspondante

`htmlentities($str)`

Renvoie une chaîne où tous les caractères ayant une entité HTML correspondante sont remplacés par celle-là

Exemple

```
<body>
<?php
$par = 'une chaine qui doit aller dans un <p>';
$attr =  'une chaîne qui doit aller dans un "attribut HTML"';
$str = 'une chaîne contenant des symboles qui ont une entité associée,
mais ne sont pas des symboles spéciaux HTML';

/*$par = htmlspecialchars($par);
$attr = htmlspecialchars($attr);
$str = htmlentities($str);*/
?>

<p> voici <?php echo $par?> </p>
<abbr title="<?php echo $attr?>"> Survole moi! </abbr>
<p> voici <?php echo $str?> </p>
</body>
```

http://localhost/.../04-htmlent.php

view source

04-htmlent.php

view source