# BDW - Programmation web - Bases du PHP

Fabien Duchateau

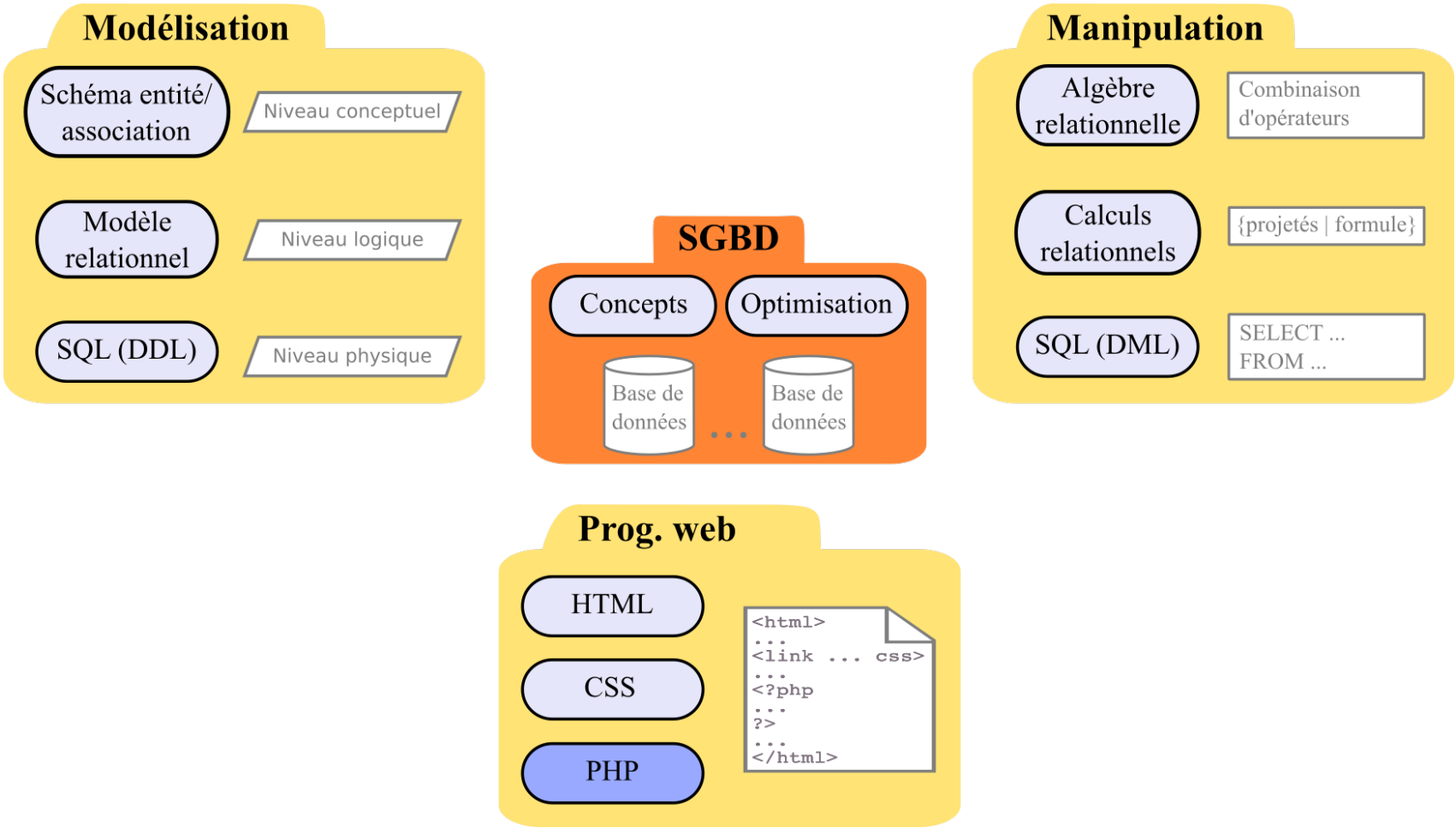
*fabien.duchateau [at] univ-lyon1.fr*

Université Claude Bernard Lyon 1

2023 - 2024


https://perso.liris.cnrs.fr/fabien.duchateau/BDW/

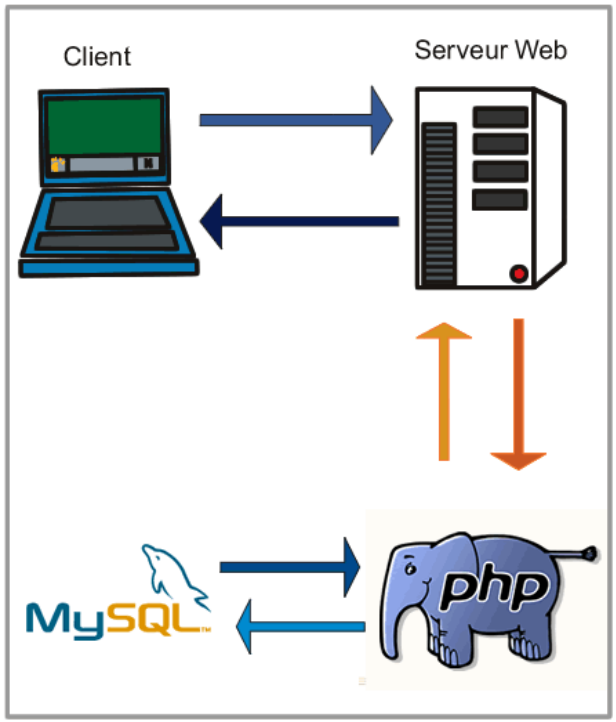
# Positionnement dans BDW




---

Ces diapositives utilisent le genre féminin (e.g., chercheuse, développeuses) plutôt que l'écriture inclusive (moins accessible, moins concise, et pas totalement inclusive)

# Rappels




HTML pour le contenu, CSS pour la mise en page/forme, et **PHP pour l'aspect dynamique (e.g., interactions avec un SGBD)**

---

http://www.phpdebutant.org/

# Rappels

Une page web dynamique est générée à la volée par un serveur

HTML et CSS insuffisants pour des besoins comme :

- Manipulation de bases de données
- Interactions avec le système de fichiers
- Utilisation de bibliothèques logicielles
- Plus généralement pour des traitements complexes

# Rappels

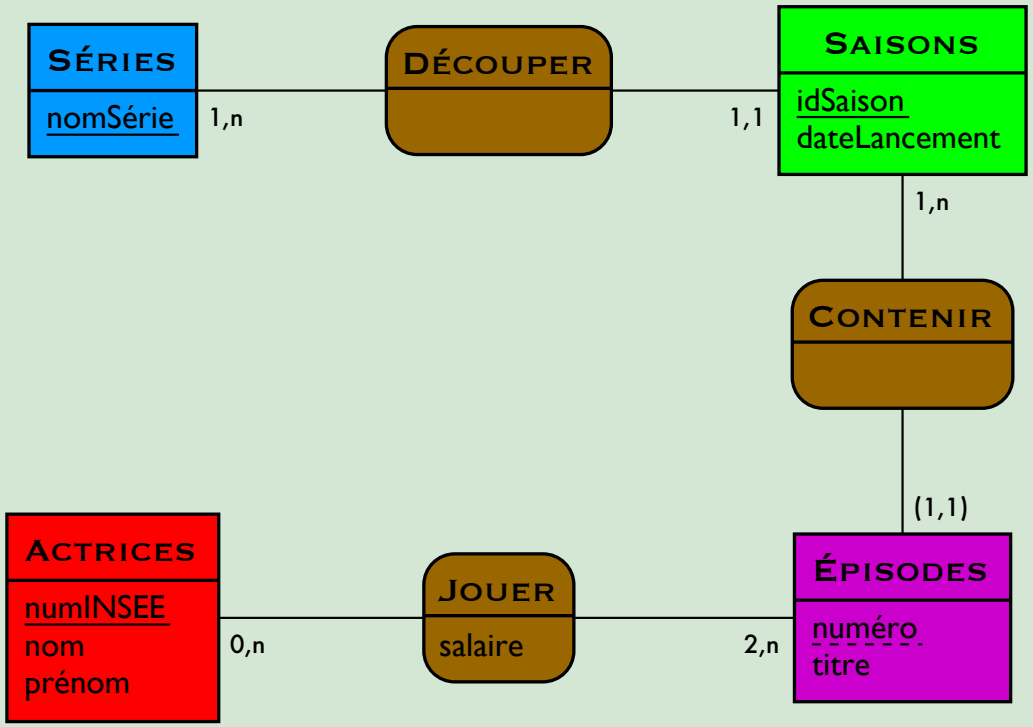
Une page web dynamique est générée à la volée par un serveur

HTML et CSS insuffisants pour des besoins comme :

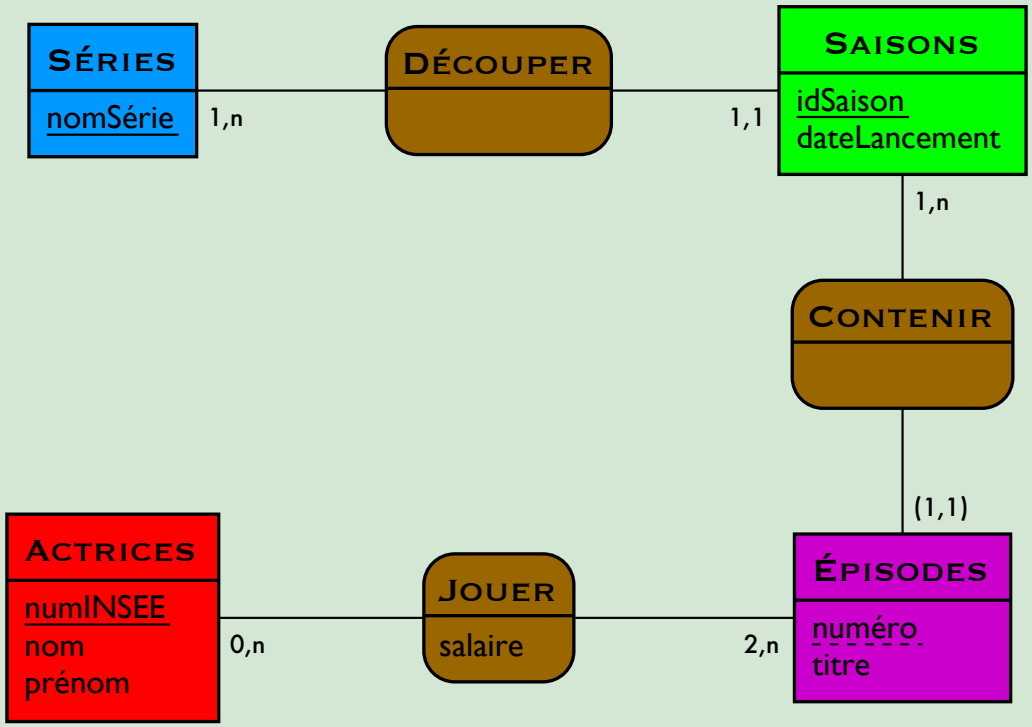
- Manipulation de bases de données
- Interactions avec le système de fichiers
- Utilisation de bibliothèques logicielles
- Plus généralement pour des traitements complexes

Ce sont quelques unes des fonctionnalités proposées par un langage de script comme PHP

# Exemple sur les séries - modèles de données



# Exemple sur les séries - modèles de données

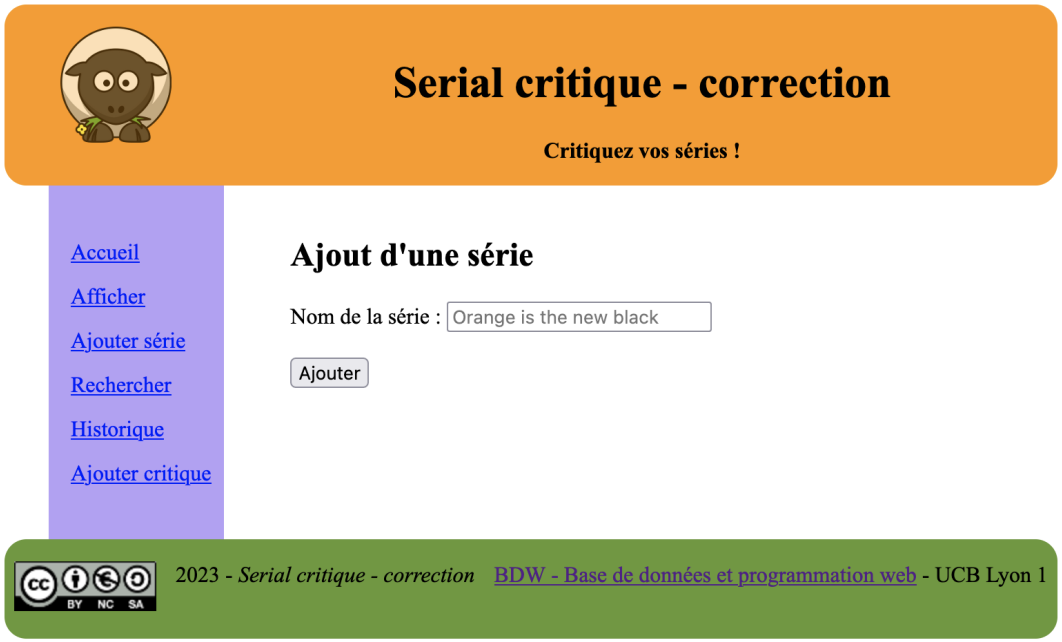



SÉRIES (nomSérie)
SAISONS (idSaison, dateLancement, #nomSérie)
ÉPISODES (numéro, #idSaison, titre)
ACTRICES (numINSEE, nom, prénom)
JOUER (#numéro, #idSaison, #numINSEE, salaire)

# Exemple sur les séries - fonctionnalités

Réalisation d'un site web pour :

- Afficher toutes les séries
- Ajouter une série
- Rechercher une actrice



# Généralités sur PHP

PHP pour *PHP : Hypertext Preprocessor* (acronyme récursif) :

- Origine : 1994 (Rasmus Lerdorf)
- Versions stables `7.4` et `8.1`
- PHP Licence (code ouvert, restriction sur le nom d'un produit dérivé)
- 80% du web tourne avec PHP (dont Wikipedia, Facebook)
- Extension d'un fichier PHP : `.php`


---

http://www.php.net/

http://fr.wikipedia.org/wiki/PHP

http://fr.wikibooks.org/wiki/Programmation_PHP

http://www.w3schools.com/php/

http://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql

# Généralités sur PHP (2)

- Un fichier PHP est le code source d'un programme
- L'utilisation de PHP en programmation web a pour but de générer une page HTML
- Mais possibilité d'utiliser PHP en ligne de commande ou pour générer une GUI (avec GTK)
- PHP est un langage proche du C, mais interprété
- Paradigmes de programmation : procédural, fonctionnel, impératif, orienté objet ou réflexif

---

http://php.net/manual/en/langref.php
http://php.net/manual/en/features.commandline.php

# Généralités sur PHP (2)

- Un fichier PHP est le code source d'un programme
- L'utilisation de PHP en programmation web a pour but de générer une page HTML
- Mais possibilité d'utiliser PHP en ligne de commande ou pour générer une GUI (avec GTK)
- PHP est un langage proche du C, mais interprété
- Paradigmes de programmation : procédural, fonctionnel, impératif, orienté objet ou réflexif

Dans notre contexte, utilisation de PHP avec le paradigme procédural et interprété à la volée

---

http://php.net/manual/en/langref.php
http://php.net/manual/en/features.commandline.php

# Plan

Syntaxe

Structures de contrôle

Fonctions

# Imbrication de HTML et PHP

Dans un fichier PHP, deux types de "zones" :

- Zones délimitées par <?php et ?> :
  - contient du code PHP à exécuter (par le serveur)
  - génère du texte intégré au contenu HTML
- Zones à l'extérieur de <?php et ?> :
  - contient du texte et des balises HTML
  - directement recopié dans le contenu HTML généré

```
<body>
<p>Ici du texte HTML
<?php
  echo ' et la suite affichée par PHP';
?>
</p>
```

# Imbrication de HTML et PHP

Dans un fichier PHP, deux types de "zones" :

- Zones délimitées par <?php et ?> :
  - contient du code PHP à exécuter (par le serveur)
  - génère du texte intégré au contenu HTML
- Zones à l'extérieur de <?php et ?> :
  - contient du texte et des balises HTML
  - directement recopié dans le contenu HTML généré

```
<body>
<p>Ici du texte HTML
<?php
  echo ' et la suite affichée par PHP';
?>
</p>
```

# Imbrication de HTML et PHP

Dans un fichier PHP, deux types de "zones" :

- Zones délimitées par <?php et ?> :
  - contient du code PHP à exécuter (par le serveur)
  - génère du texte intégré au contenu HTML
- Zones à l'extérieur de <?php et ?> :
  - contient du texte et des balises HTML
  - directement recopié dans le contenu HTML généré

```
<body>
<p>Ici du texte HTML
<?php
 echo ' et la suite affichée par PHP';
?>
</p>
```

Ici du texte HTML et la suite affichée par PHP

# Imbrication de HTML et PHP (2)

Deux méthodes pour générer du contenu :

- Dans une zone PHP, utiliser l'instruction `print` ou `echo` :
  - ce texte est ajouté à la suite du contenu HTML déjà généré
- Mettre du texte à l'extérieur de `<?php` et `?>`

```
<?php
$a = 'Délaisse les grandes routes, ';
$b = '(Pythagore)';
echo $a;
?>
prends les sentiers
<?php print $b; ?>
```

# Imbrication de HTML et PHP (2)

Deux méthodes pour générer du contenu :

- Dans une zone PHP, utiliser l'instruction `print` ou `echo` :
  - ce texte est ajouté à la suite du contenu HTML déjà généré
- Mettre du texte à l'extérieur de `<?php` et `?>`

```
<?php
$a = 'Délaisse les grandes routes, ';
$b = '(Pythagore)';
echo $a;
?>
prends les sentiers
<?php print $b; ?>
```

Délaisse les grandes routes, prends les sentiers (Pythagore)

# Commentaires

Commentaire uniligne :

```
// commentaire
```

Commentaire multiligne :

```
/* commentaire
sur 2 lignes */
```

# Variables

- PHP est un langage à typage faible
- Le nom d'une variable commence par un **$**
- Affectation :

```
$nom_var = valeur ;
```

- Les variables ne sont pas explicitement déclarées comme en C (une variable existe dès qu'une valeur lui est affectée)

```
$compteur = 1 ;
$chaine = 'BDW' ;
```

# Types de variables

- `boolean`, `integer`, `double`, `string`, `array`, `object` (instance de classe), `resource` (type abstrait, renvoyé par des fonctions), `NULL` (absence de valeur)
- Une variable peut contenir un nombre, une chaîne de caractères, un booléen ou un tableau sans préciser le type
- PHP effectue automatiquement des conversions de type si nécessaire

---

http://php.net/manual/en/language.types.php

# Bonnes pratiques

- Langage sensible à la casse (`$var` est différent de `$Var`)
- Nommage d'une variable : le symbole dollar, puis une lettre, puis une combinaison de lettre/chiffre/underscore
- Visibilité d'une variable : dans le bloc où elle est déclarée (fonction, classe, bloc d'instructions)

```
$nom_var = 1 ; // correct
$2_var = 2 ; // incorrect
```

# Chaînes de caractères

- Délimitées par des guillemets simples (`'chaine'`) ou doubles (`"chaine"`)
- Dans une chaîne avec guillemets doubles, une variable est remplacée par sa valeur (convertie en chaîne de caractères) et les séquences d'échappement sont interprétées
- Concaténation de chaînes avec **le point .**

```
$acro = 'BD';
echo '$acro : blu-ray disk<br>';
echo "$acro : base de données<br>";
echo $acro . ' : bande dessinée';
```

---

http://php.net/manual/en/language.types.string.php

# Chaînes de caractères

- Délimitées par des guillemets simples (`'chaine'`) ou doubles (`"chaine"`)
- Dans une chaîne avec guillemets doubles, une variable est remplacée par sa valeur (convertie en chaîne de caractères) et les séquences d'échappement sont interprétées
- Concaténation de chaînes avec **le point .**

```
$acro = 'BD';
echo '$acro : blu-ray disk<br>';
echo "$acro : base de données<br>";
echo $acro . ' : bande dessinée';
```

```
$acro : blu-ray disk
BD : base de données
BD : bande dessinée
```

http://php.net/manual/en/language.types.string.php

# Tableaux

- Carte ordonnée, qui associe une clé à une valeur
  - une clé est un entier ou une chaîne de caractères
  - une valeur est d'un type quelconque (y compris un tableau)
- Pour créer un tableau vide :

```
$mon_tableau = array() ;
```

- Pour affecter `valeur` à la case identifiée par *clé* :

```
$mon_tableau[clé] = valeur ;
```

- Pour ajouter une case avec la valeur `valeur` :

```
$mon_tableau[] = valeur ;
```

---

http://php.net/manual/en/language.types.array.php

# Tableaux (2)

```
fabien@laptop :-$ php -a
Interactive shell

php > $responsables = array();
php > $responsables['LIFIHM'] = 'Stephanie JD';
php > $responsables['BDW'] = "Fabien D";
php > print_r($responsables);
Array
(
        [LIFIHM] => Stephanie JD
        [BDW] => Fabien D
)
php >
```

*Un exemple de tableau avec PHP en mode console ("interactive shell"). Le mode interactif seul ("interactive mode enabled") permet seulement d'interpréter un contenu à la volée*

# Opérateurs

- Un opérateur prend de une à trois valeurs (expressions) et retourne une valeur
  - unaire (e.g., négation, incrément de 1)
  - binaire (e.g., addition, égalités)
  - ternaire (l'unique opérateur conditionnel)

- Les opérateurs s'évaluent selon l'ordre de précédence (cf URL)

---

http://php.net/manual/en/language.operators.php
http://php.net/manual/en/language.operators.precedence.php

# Quelques opérateurs

Arithmétiques :

- + (addition), - (soustraction), * (multiplication), / (division), % (modulo), ++ (incrément), -- (décrément)

De comparaison :

- == (égalité avec conversion de type), === (égalité sans conversion de type), < (inférieur strict), <= (inférieur large), >, >=, != ou <> (différence)

Logiques :

- `and`, && (et), `or`, || (ou), `xor` (ou exclusif), ! (négation)

---

http://php.net/manual/en/language.operators.arithmetic.php
http://php.net/manual/en/language.operators.comparison.php
http://php.net/manual/en/language.operators.logical.php

# Quelques opérateurs (2)

De type (classe par extension) :

- `instanceof` (déterminer si une variable / objet est de la classe mentionnée)

```
$nom_var instanceof uneClasse ; // true or false
```

De tableau :

- + (union de 2 tableaux), == (même paires clé/valeur), === (même paires clé/valeur, dans le même ordre et de même type), != (au moins une paire clé/valeur différente), !== (au moins un triplet clé/valeur/type différent), etc.

---

http://php.net/manual/en/language.operators.type.php
http://php.net/manual/en/language.operators.array.php

# Fonctions

- Une fonction accepte 0 ou plusieurs paramètres, et calcule une valeur de retour
- Possibilité de définir ses propres fonctions
- De nombreuses fonctions, ici celles qui concernent les variables / types

---

http://php.net/manual/en/ref.var.php

# Quelques fonctions de type

- Afficher le type d'une variable :

```
gettype($nom_var) ; // boolean, string, array, etc.
```

- Afficher le type et le contenu d'une variable :

```
var_dump($nom_var) ;
```

- Fonctions `intval`, `strval`, `floatval` ou `settype` pour une conversion de type :

```
strval($nom_var) ;
settype($nom_var, 'string') ;
```

# Quelques fonctions de variable

- Tester si une variable *$var* existe :

```
isset($var)
```

- Tester si une variable *$var* est nulle :

```
empty($var)
```

```
$chiffre = 4;
$chaine = 'hello';
print isset($chiffre); // retourne 1
print isset($chaine); // retourne 1
print isset($jexistepas); // aucun retour
print empty($chaine); // aucun retour
print empty($jexistepas); // retourne 1
```

# Constantes

- Un nom donné à une valeur fixée :
  - créée avec la fonction `define` (en général, nom en majuscules)
  - utilisée par appel à son nom (pas de dollar)
- Fonction `defined` pour savoir si une constante est définie
- Constantes prédéfinies (e.g., `__DIR__` pour le répertoire courant)

```
define('MAX_COLONNES', 5);
define('NOM_CONTACT', 'qqun@ici.net');
echo 'Contact : '.NOM_CONTACT; // Contact : qqun@ici.net
echo defined('MAX_COLONNES'); // retourne 1
```

---

http://php.net/manual/en/language.constants.php
http://php.net/manual/en/language.constants.predefined.php

# Syntaxe générale

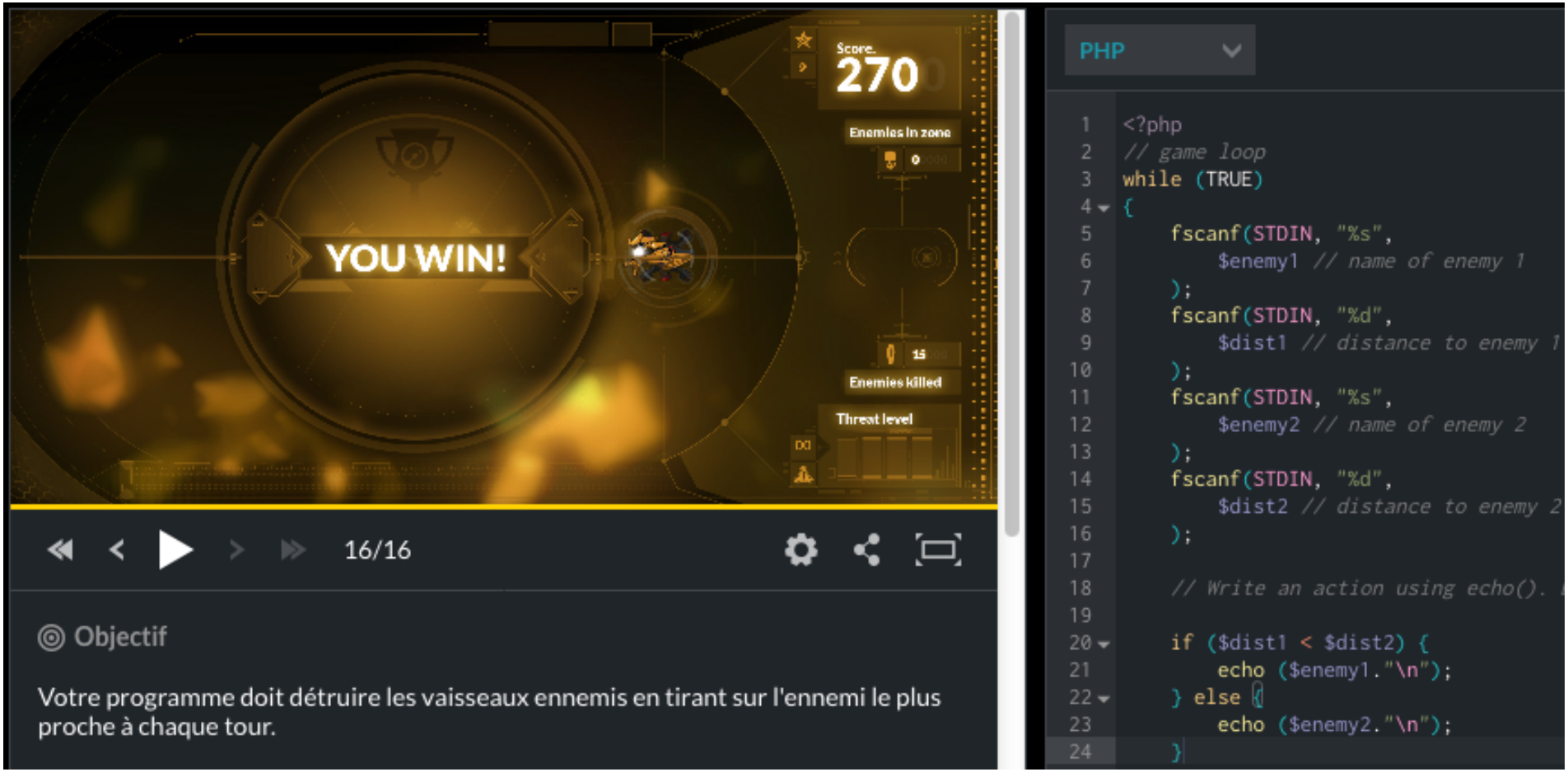
Un **programme**, c'est une suite d'instructions (e.g., assignation, appel de fonction, instruction conditionnelle)

Une **instruction** se termine par un point virgule

Un **bloc d'instructions** est placé entre accolades

```
{
  define('MAX_COLONNES', 5);
  $col_courante = 2;
  $nb_col_restantes = MAX_COLONNES - $col_courante;
  echo "Il reste $nb_col_restantes colonnes à remplir";
}
```

# En résumé




---

https://www.codingame.com/
https://www.codecademy.com/catalog/language/php

# Plan

Syntaxe

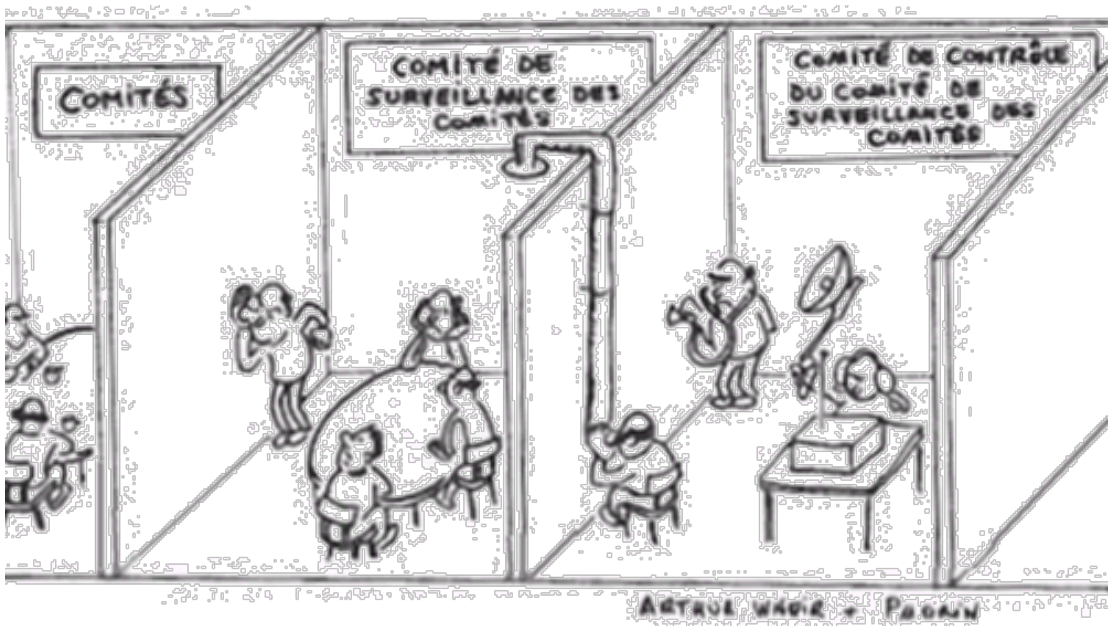
Structures de contrôle

Fonctions

# Généralités

Les structures de contrôle permettent les traitements :

- Conditionnels
- Itératifs (boucles)




http://www.php.net/manual/en/language.control-structures.php

# Structure IF...ELSE

Structure conditionnelle similaire à celle de C :

- Si *conditions* renvoient TRUE, exécution des instructions dans le bloc du `if`
- Sinon, on exécute les instructions dans le bloc du `else`
- Instruction `else` facultative

```
if(conditions) {
  <instructions pour conditions vérifiées>
}
else {
  <instructions pour conditions non vérifiées>
}
```

# Un exemple de structure IF...ELSE

```php
<?php // exemple if
$chaine = 'hello';
if(isset($chaine)) {
  print $chaine;
}
else {
  print "La variable n'existe pas !";
}
?>
```

# Un exemple de structure IF...ELSE

```php
<?php // exemple if
$chaine = 'hello';
if(isset($chaine)) {
  print $chaine;
}
else {
  print "La variable n'existe pas !";
}
?>
```

*Résultat du script : hello*

# Structure IF...ELSEIF...ELSE

En cas de tests multiples, on peut ajouter une ou plusieurs structures conditionnelles `elseif`

```
if(conditions_1) {
    <instructions pour conditions_1 vérifiées>
}
elseif(conditions_2) {
    <instructions pour conditions_2 vérifiées>
 }
else {
    <instructions si aucune condition vérifiée>
}
```

http://fr2.php.net/manual/fr/control-structures.switch.php

# Un exemple de structure IF...ELSEIF...ELSE

```php
<?php // exemple elseif
$chaine = '';
if(! isset($chaine)) {
  print "La variable chaine n'existe pas";
}
elseif(empty($chaine)) {
  print "La variable chaine existe mais sans valeur";
}
else {
  print "La variable chaine existe et vaut ".$chaine;
}
?>
```

# Un exemple de structure IF...ELSEIF...ELSE

```php
<?php // exemple elseif
$chaine = '';
if(! isset($chaine)) {
  print "La variable chaine n'existe pas";
}
elseif(empty($chaine)) {
  print "La variable chaine existe mais sans valeur";
}
else {
  print "La variable chaine existe et vaut ".$chaine;
}
?>
```

*Résultat du script : La variable chaine existe mais sans valeur*

# Structure SWITCH

Le `switch` est une alternative aux multiples `elseif`, quand une liste de valeurs est connue (e.g., un menu)

- Un `case` s'exécute si sa valeur est celle de *expression*, et le code est exécuté soit jusque la fin du bloc soit jusqu'à une instruction `break`
- Le cas `default` est exécuté si aucun autre `case` ne l'est

```
switch(expression) {
  case valeur_1 :
    <instructions pour valeur_1>
    break ;
  ...
  default :
    <instructions par défaut>
}
```

# Un exemple de structure SWITCH

```php
<?php // exemple switch
$choix_menu = 2;
switch($choix_menu) {
  case 1:
    echo "Entrée; Plat; Boisson";
    break;
  case 2:
    echo "Dessert;";
  case 3:
    echo "Plat; Boisson";
    break;
}
?>
```

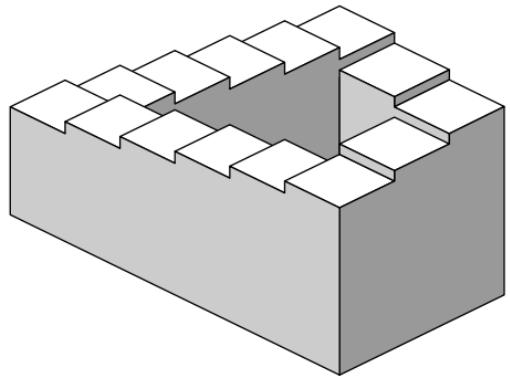
# Un exemple de structure SWITCH

```php
<?php // exemple switch
$choix_menu = 2;
switch($choix_menu) {
  case 1:
    echo "Entrée; Plat; Boisson";
    break;
  case 2:
    echo "Dessert;";
  case 3:
    echo "Plat; Boisson";
    break;
}
?>
```

*Résultat du script : Dessert ; Plat ; Boisson*

# Les boucles

- Permettent de répéter un bloc d'instructions tant qu'une condition est vérifiée
- Utile pour parcourir un tableau, ou lire un fichier, etc.
- Trois types de boucles en PHP :
  - `while`
  - `for`
  - `foreach`



http://fr.wikipedia.org/wiki/Escalier_de_Penrose

# Boucle WHILE

Une boucle `while` ("tant que") exécute les *instructions* tant que la *condition* est vérifiée

```
while(condition) {
  <instructions>
}
```

# Boucle WHILE

Une boucle `while` ("tant que") exécute les *instructions* tant que la *condition* est vérifiée

while(*condition*) {
  <instructions>
}

```php
<?php // exemple while
$n = 2;
while($n < 10) {
  print "$n ";
  $n = $n + 3;
}
?>
```

# Boucle WHILE

Une boucle `while` ("tant que") exécute les *instructions* tant que la *condition* est vérifiée

```
while(condition) {
  <instructions>
}
```

```php
<?php // exemple while
$n = 2;
while($n < 10) {
  print "$n ";
  $n = $n + 3;
}
?>
```

*Résultat du script : 2 5 8*

# Boucle FOR

Une boucle `for` ("pour") exécute les *instructions* tant que la *condition* est vérifiée. L'*initialisation* est exécutée en début de boucle. Si la condition est vérifiée, *instructions* et *itération* sont exécutées, puis si la condition est à nouveau vérifiée, *instructions* et *itération* sont exécutées, ...

```
for(initialisation ; condition ; itération) {
  <instructions>
}
```

# Boucle FOR

Une boucle `for` ("pour") exécute les *instructions* tant que la *condition* est vérifiée. L'*initialisation* est exécutée en début de boucle. Si la condition est vérifiée, *instructions* et *itération* sont exécutées, puis si la condition est à nouveau vérifiée, *instructions* et *itération* sont exécutées, ...

```
for(initialisation ; condition ; itération) {
  <instructions>
}
```

```php
<?php // exemple for
for($n = 3; $n <= 9; $n+=2) {
  print "$n ";
}
?>
```

# Boucle FOR

Une boucle `for` ("pour") exécute les *instructions* tant que la *condition* est vérifiée. L'*initialisation* est exécutée en début de boucle. Si la condition est vérifiée, *instructions* et *itération* sont exécutées, puis si la condition est à nouveau vérifiée, *instructions* et *itération* sont exécutées, ...

```
for(initialisation ; condition ; itération) {
  <instructions>
}
```

```php
<?php // exemple for
for($n = 3; $n <= 9; $n+=2) {
  print "$n ";
}
?>
```

*Résultat du script : 3 5 7 9*

# Boucle FOREACH

Une boucle `foreach` ("pour chaque") permet de parcourir des tableaux vus comme un ensemble de valeurs

```
foreach($tableau as $valeur) {
  <instructions>
}
```

# Boucle FOREACH

Une boucle `foreach` ("pour chaque") permet de parcourir des tableaux vus comme un ensemble de valeurs

```
foreach($tableau as $valeur) {
  <instructions>
}
```

```php
<?php // exemple foreach
$tab = array('foo', 'bar', 3);
foreach($tab as $val) {
  print "$val ";
}
?>
```

# Boucle FOREACH

Une boucle `foreach` ("pour chaque") permet de parcourir des tableaux vus comme un ensemble de valeurs

```
foreach($tableau as $valeur) {
  <instructions>
}
```

```php
<?php // exemple foreach
$tab = array('foo', 'bar', 3);
foreach($tab as $val) {
  print "$val ";
}
?>
```

*Résultat du script : foo bar 3*

# Boucle FOREACH (2)

La boucle `foreach` permet aussi de parcourir des tableaux associatifs vus comme un ensemble de paires clé/valeur

```
foreach($tableau_asso as $cle ⇒ $valeur) {
  <instructions>
}
```

# Boucle FOREACH (2)

La boucle `foreach` permet aussi de parcourir des tableaux associatifs vus comme un ensemble de paires clé/valeur

```
foreach($tableau_asso as $cle ⇒ $valeur) {
  <instructions>
}
```

```php
<?php // exemple foreach associatif
$tab_asso = array("nom" => "Dupont", "age" => 21);
foreach($tab_asso as $cle => $val) {
  print "$cle : $val \t";
}
?>
```

# Boucle FOREACH (2)

La boucle `foreach` permet aussi de parcourir des tableaux associatifs vus comme un ensemble de paires clé/valeur

```
foreach($tableau_asso as $cle ⇒ $valeur) {
  <instructions>
}
```

```php
<?php // exemple foreach associatif
$tab_asso = array("nom" => "Dupont", "age" => 21);
foreach($tab_asso as $cle => $val) {
  print "$cle : $val \t";
}
?>
```

*Résultat du script : nom : Dupont age : 21*

# Syntaxes alternatives

Une syntaxe alternative permet de clarifier le code (en particulier quand le PHP est imbriqué avec du HTML)

- Valable pour les instructions `if`, `while`, `for`, `foreach`, `switch`
- Les accolades du bloc sont remplacées par un `":"` et un `"end"` (adapté à l'instruction)

```
<?php if (condition) : ?>
Si la condition est vérifiée, ce texte s'affiche
<?php endif ; ?>
```

http://php.net/manual/en/control-structures.alternative-syntax.php

# Exemples de syntaxes alternatives

```
1 <?php if ($a == 1): ?>
2 a vaut 1
3 <?php else: ?>
4 a ne vaut pas 1
5 <?php endif; ?>
```

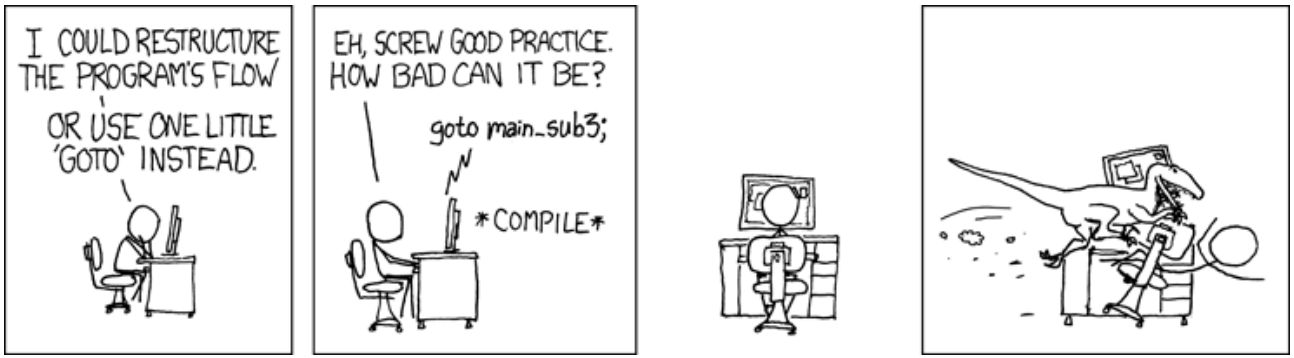
*Script qui imbrique PHP et HTML, avec la syntaxe alternative*

```
1 <?php if ($a == 1) { ?>
2 a vaut 1
3 <?php } else { ?>
4 a ne vaut pas 1
5 <?php } ?>
```

*Même script, avec la syntaxe normale (accolades)*

# Des structures de contrôle moins fréquentes

- `goto` permet de sauter directement à un autre endroit du code, représenté par un label
- `continue` s'utilise dans une boucle pour arrêter l'itération en cours et passer à la suivante
- `break` permet de quitter la boucle courante




---

http://php.net/manual/en/control-structures.goto.php
http://php.net/manual/en/control-structures.continue.php
http://php.net/manual/en/control-structures.break.php
http://xkcd.com/292

# En résumé

- Syntaxe proche du C pour les structures conditionnelles
- Boucles traditionnelles (`while` et `for`) et une boucle spécifique (`foreach`)
- Des syntaxes alternatives pour clarifier le code


http://vidberg.blog.lemonde.fr/

# Plan

Syntaxe

Structures de contrôle

Fonctions

# Généralités

Une fonction est un ensemble d'instructions réutilisable :

- Elle prend en entrée $0$ ou plusieurs paramètres
- Elle retourne éventuellement une valeur (conseillé, ne serait-ce qu'un booléen TRUE/FALSE indiquant le succès/échec de l'exécution de la fonction)
- La **signature** d'une fonction inclut son nom, ses paramètres et son type de retour

Quand utiliser des fonctions ?

- Traitement récurrent (affichage, calcul, etc.)
- Connexion et déconnexion à la BD
- ...

# Définition d'une fonction

En PHP, une fonction se définit par le mot clé `function` :

- *nom_fonction* = le nom de la fonction
- *$param1*, *$param2* = des paramètres
- $valeur = valeur de retour, introduite par le mot clé `return`

```
function nom_fonction($param1, $param2, ...) {
  <instructions>
  return $valeur;
}
```

---

http://php.net/manual/en/language.functions.php

# Exemple de fonction

```php
<?php // fonction somme
function sommer($a, $b) {
  return $a + $b;
}
echo sommer(1, 2);
?>
```

*Une fonction qui somme deux nombres passés en paramètres*

# Notions avancées

- Les paramètres d'une fonction peuvent avoir une valeur par défaut (et devenir optionnel lors de l'appel à la fonction)
- Le type de retour d'une fonction peut être précisé
- Support de fonctions variables (e.g., pour implémenter des callbacks)
- Support de fonctions anonymes (closures)
- L'existence d'une fonction peut-être vérifiée par la fonction `function_exists()`

---

http://php.net/manual/en/functions.arguments.php
http://php.net/manual/en/functions.returning-values.php
http://php.net/manual/en/functions.variable-functions.php
http://php.net/manual/en/function.function-exists.php

# Exemples de notions avancées

```php
<?php // fonction somme (avec retour typé)
function sommerF($a, $b): float {
  return $a + $b;
}
var_dump(sommerF(1, 2));
?>
```

*Une fonction qui somme deux nombres passés en paramètres, mais la valeur de retour sera un `float`*

```php
<?php // vérifier l'existence d'une fonction
if(function_exists('var_dump') && function_exists('sommerF')) {
  echo "Les fonctions existent.";
  var_dump(sommerF(1, 2));
} else {
  echo "L'une des deux fonctions n'existe pas !";
}
?>
```

*Un script qui vérifie l'existence de fonctions avant de les appeler*

# Un moment de réflexion

## Écrire une fonction pour colorer des notes stockées dans un tableau (vert pour les notes supérieures à $10$, rouge sinon)

```
<?php

$notes = array(12, 17, 8, 10, 14, 3);

?>
```

# Un moment de réflexion

## Écrire une fonction pour colorer des notes stockées dans un tableau (vert pour les notes supérieures à 10, rouge sinon)

```
<?php
function colorier($note) { // un paramètre de fonction : $note




} // fin de la fonction

$notes = array(12, 17, 8, 10, 14, 3);



?>
```

# Un moment de réflexion

## Écrire une fonction pour colorer des notes stockées dans un tableau (vert pour les notes supérieures à 10, rouge sinon)

```
<?php
function colorier($note) { // un paramètre de fonction : $note
    if($note > 10)

    else // note inférieure ou égale à 10

} // fin de la fonction

$notes = array(12, 17, 8, 10, 14, 3);



?>
```

# Un moment de réflexion

## Écrire une fonction pour colorer des notes stockées dans un tableau (vert pour les notes supérieures à 10, rouge sinon)

```
<?php
function colorier($note) { // un paramètre de fonction : $note
    if($note > 10)
        echo '<span style="color : #00FF00">' .$note. ' </span>';
    else // note inférieure ou égale à 10
        echo '<span style="color : #FF0000">' .$note.' </span>';
} // fin de la fonction

$notes = array(12, 17, 8, 10, 14, 3);



?>
```

# Un moment de réflexion

## Écrire une fonction pour colorer des notes stockées dans un tableau (vert pour les notes supérieures à 10, rouge sinon)

```
<?php
function colorier($note) { // un paramètre de fonction : $note
    if($note > 10)
        echo '<span style="color : #00FF00">' .$note. ' </span>';
    else // note inférieure ou égale à 10
        echo '<span style="color : #FF0000">' .$note.' </span>';
} // fin de la fonction

$notes = array(12, 17, 8, 10, 14, 3);
foreach($notes as $note) { // parcours du tableau


}
?>
```

# Un moment de réflexion

## Écrire une fonction pour colorer des notes stockées dans un tableau (vert pour les notes supérieures à 10, rouge sinon)

```
<?php
function colorier($note) { // un paramètre de fonction : $note
    if($note > 10)
        echo '<span style="color : #00FF00">' .$note. ' </span>';
    else // note inférieure ou égale à 10
        echo '<span style="color : #FF0000">' .$note.' </span>';
} // fin de la fonction

$notes = array(12, 17, 8, 10, 14, 3);
foreach($notes as $note) { // parcours du tableau
    colorier($note); // appel de la fonction pour chaque note
}
?>
```

# Un moment de réflexion

## Écrire une fonction pour colorer des notes stockées dans un tableau (vert pour les notes supérieures à 10, rouge sinon)

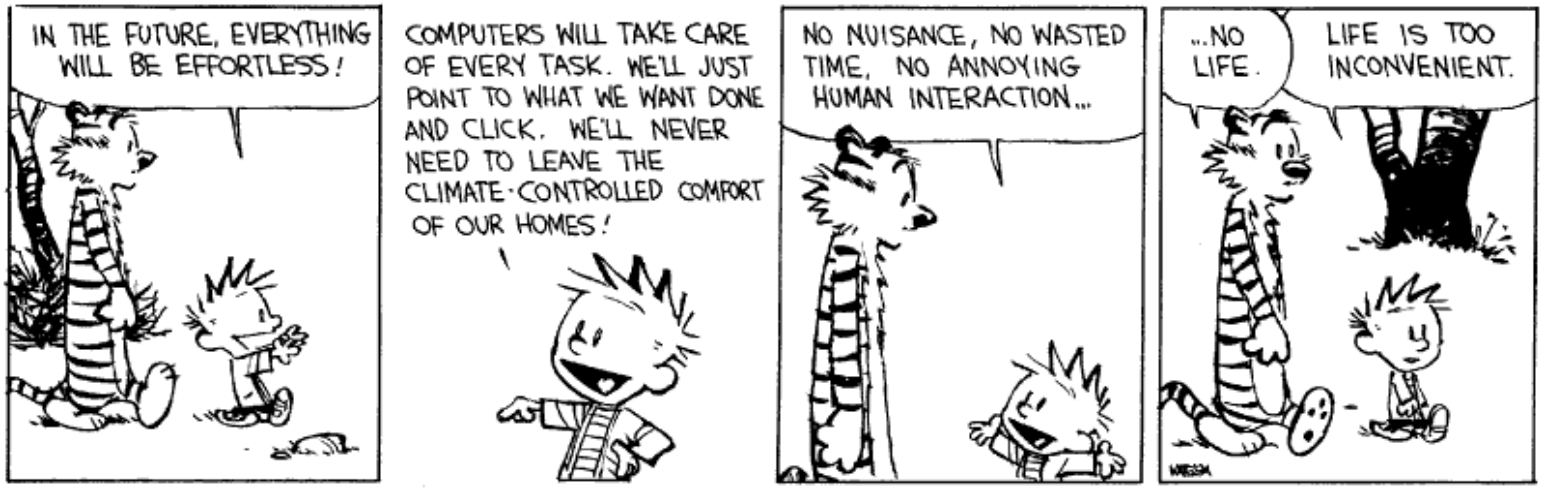
```
<?php
function colorier($note) { // un paramètre de fonction : $note
    if($note > 10)
        echo '<span style="color : #00FF00">' .$note. ' </span>';
    else // note inférieure ou égale à 10
        echo '<span style="color : #FF0000">' .$note.' </span>';
} // fin de la fonction

$notes = array(12, 17, 8, 10, 14, 3);
foreach($notes as $note) { // parcours du tableau
    colorier($note); // appel de la fonction pour chaque note
} // résultat : "12  17  8  10  14  3"
?>
```

# Fonctions disponibles en PHP

En plus des fonctions que vous pouvez créer, le noyau (core) de PHP intègre de nombreuses fonctions, par exemple :

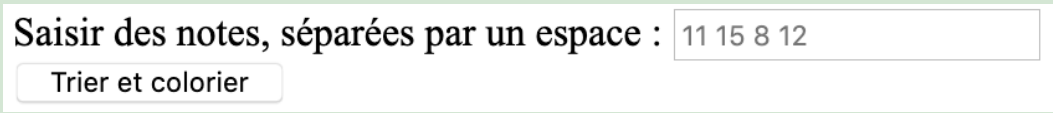
- Manipulation de tableaux, de chaînes
- Manipulation du système de fichiers
- Manipulation de bases de données




Calvin et Hobbes (Bill Watterson)

# Un moment de réflexion

## Écrire une page qui permet de saisir des notes dans un champ texte, de trier ces notes et de les afficher coloriées

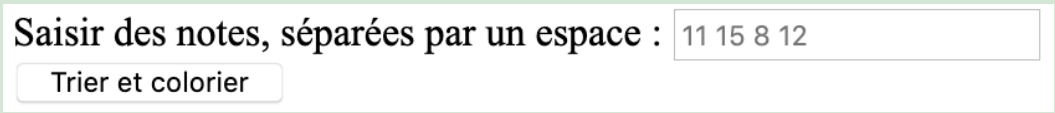



---

http://www.php.net/manual/en/function.explode.php
http://www.php.net/manual/en/function.sort.php
http://www.php.net/manual/en/function.is-numeric.php

# Un moment de réflexion

## Écrire une page qui permet de saisir des notes dans un champ texte, de trier ces notes et de les afficher coloriées



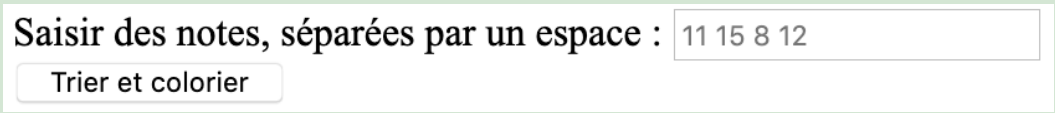

- Formulaire de saisie des notes (HTML)

---

http://www.php.net/manual/en/function.explode.php
http://www.php.net/manual/en/function.sort.php
http://www.php.net/manual/en/function.is-numeric.php

# Un moment de réflexion

## Écrire une page qui permet de saisir des notes dans un champ texte, de trier ces notes et de les afficher coloriées



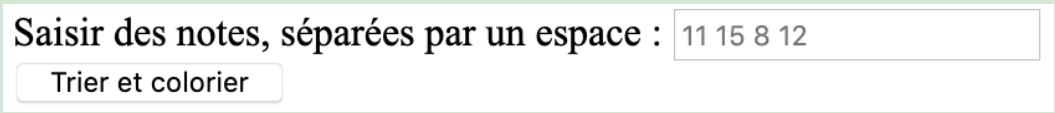

- Formulaire de saisie des notes (HTML)
- Script pour traiter le formulaire (PHP) :

---

http://www.php.net/manual/en/function.explode.php
http://www.php.net/manual/en/function.sort.php
http://www.php.net/manual/en/function.is-numeric.php

# Un moment de réflexion

## Écrire une page qui permet de saisir des notes dans un champ texte, de trier ces notes et de les afficher coloriées



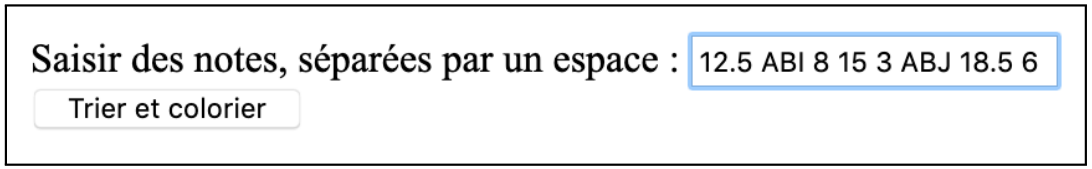

- Formulaire de saisie des notes (HTML)
- Script pour traiter le formulaire (PHP) :
  - Récupération des notes (sous forme de chaîne)
  - Conversion de la chaîne en tableau
  - Tri du tableau
  - Affichage (colorié) des notes (numériques)

---

http://www.php.net/manual/en/function.explode.php
http://www.php.net/manual/en/function.sort.php
http://www.php.net/manual/en/function.is-numeric.php

# Un moment de réflexion (2)

```
1 <form method="post">
2   <label for="idTab">Saisir des notes, séparées par un espace :</label>
3   <input type="text" id="idTab" name="tab" required placeholder="11 15 8 12"><br>
4   <input type="submit" name="bValider" value="Trier et colorier">
5 </form>
```



# Un moment de réflexion (2)

```html
<form method="post">
  <label for="idTab">Saisir des notes, séparées par un espace :</label>
  <input type="text" id="idTab" name="tab" required placeholder="11 15 8 12"><br>
  <input type="submit" name="bValider" value="Trier et colorier">
</form>
```

Saisir des notes, séparées par un espace : 12.5 ABI 8 15 3 ABJ 18.5 6

Trier et colorier

```php
<?php
function colorier($note) {
  $color = "#FF0000";  // couleur rouge par défaut
  if($note > 10)
    $color = "#00FF00"; // couleur vert si note > 10
  echo '<span style="color: ' . $color . '; margin-right:1em;">' .$note. '</span>';
}

if(isset($_POST["bValider"])) { // formulaire soumis
  $chNotes = $_POST["tab"];
  $tabNotes = explode(" ", $chNotes);
  sort($tabNotes);
  foreach($tabNotes as $note) {
    if(is_numeric($note))
      colorier($note);
  }
}
?>
```

Saisir des notes, séparées par un espace : 11 15 8 12

Trier et colorier

3 6 8 12.5 15 18.5

# En résumé

PHP comme langage de programmation web :

- Programmation en procédural, pour générer du HTML
- Réflexion pour bien factoriser le code en fonctions
- Limiter l'affichage de code HTML avec PHP

# En résumé

PHP comme langage de programmation web :

- Programmation en procédural, pour générer du HTML
- Réflexion pour bien factoriser le code en fonctions
- Limiter l'affichage de code HTML avec PHP

**Prochain cours : superglobales, sessions, fichiers, et bases de données**