codecademy

Conditionnels et logique en PHP

Instruction PHP else

Une instruction PHP else peut suivre un `if` bloc. Si la condition de `if` n'est pas évaluée à `TRUE` , le bloc de code suivant `else` sera exécuté.

```
$condition = FALSE;
if ($condition) {
  // This code block will not execute
} else {
  // This code block will execute
}
```

Opérateurs de comparaison PHP

Les opérateurs de comparaison PHP sont utilisés pour comparer deux valeurs et retourner `TRUE` ou `FALSE` selon la validité de la comparaison. Les opérateurs de comparaison incluent :

- identique (`===`)
- pas identique (`!==`)
- supérieur à (`>`)
- moins de (`<`)
- supérieur ou égal (`>=`)
- inférieur ou égal (`<=`)

```
// Comparison operators
1 > 3; // FALSE
3 > 1; // TRUE
250 >= 250; // TRUE
1 === 1; // TRUE
1 === 2; // FALSE
1 === "1"; // FALSE
```

PHP si les déclarations

Les instructions PHP `if` évaluent une valeur ou une expression booléenne et exécutent le bloc de code fourni si l'expression est évaluée à `TRUE` .

```
if (TRUE){
  echo "TRUE is always true";
}

$condition1 = TRUE;
if ($condition1) {
  // This code block will execute
}

$condition2 = FALSE;
if ($condition2) {
  // This code block will not execute
}
```

Instruction de commutateur PHP

Les instructions PHP `switch` fournissent une syntaxe claire pour une série de comparaisons dans lesquelles une valeur ou une expression est comparée à de nombreuses correspondances possibles et des blocs de code sont exécutés en fonction de la correspondance `case` .
En PHP, une fois qu'une correspondance `case` est rencontrée, les blocs de code de tous les cas suivants (indépendamment de la correspondance) seront exécutés jusqu'à ce qu'un `return` , `break` ou la fin de l'instruction soit atteinte. C'est ce qu'on appelle *tomber à travers* .

```
switch ($letter_grade){
  case "A":
    echo "Terrific";
    break;
  case "B":
    echo "Good";
    break;
  case "C":
    echo "Fair";
    break;
  case "D":
    echo "Needs Improvement";
    break;
  case "F":
    echo "See me!";
    break;
  default:
    echo "Invalid grade";
}
```

PHP readline()

La fonction intégrée PHP `readline()` prend une chaîne avec laquelle inviter l'utilisateur. Il attend que l'utilisateur saisisse du texte dans le terminal et renvoie cette valeur sous forme de chaîne.

```
echo "\nWhat's your name?\n";
$name = readline(">> "); // receives user
input
```

Instructions PHP elseif

Les instructions PHP `elseif` doivent être associées à une `if` instruction, mais de nombreux `elseif` s peuvent être chaînés à partir d'un seul `if` .
`elseif` s fournissent une condition supplémentaire pour vérifier (et le code correspondant à exécuter) si les instructions conditionnelles du `if` bloc et de tout s précédent `elseif` ne sont pas remplies.

```
$fav_fruit = "orange";

if ($fav_fruit === "banana"){
  echo "Enjoy the banana!";
} elseif ($fav_fruit === "apple"){
  echo "Enjoy the apple!";
} elseif ($fav_fruit === "orange"){
  echo "Enjoy the orange!";
} else {
  echo "Enjoy the fruit!";
}
// Prints: Enjoy the orange!
```

Vérité et fausseté PHP

Les valeurs PHP dans une condition seront toujours évaluées à `TRUE` ou `FALSE` . Les valeurs évaluées `TRUE` sont appelées *truey* et les valeurs évaluées `FALSE` sont appelées *falsy* .
Les fausses valeurs incluent :

- `false`
- `0`
- chaînes vides
- `null`
- `undefined`
- `NaN` .

Toutes les autres valeurs sont *véridiques* .

```
if ("What's going on?"){   // evaluates
to TRUE
  echo "Let us explain…";
}
// Prints: Let us explain…
```

Valeurs booléennes PHP

Les valeurs booléennes PHP sont soit `TRUE` ou `FALSE` , qui sont les seuls membres du `boolean` type

```
// booleans
$is_true = TRUE;
$is_false = FALSE;

echo gettype($is_true);
// Prints: boolean
echo gettype($is_false);
// Prints: boolean
```

Opérateur ternaire PHP

En PHP, l'opérateur ternaire permet une syntaxe compacte dans le cas de `if/else` décisions binaires ( ).
Il évalue une seule condition et exécute une expression et renvoie sa valeur si la condition est remplie et la seconde expression dans le cas contraire.
La syntaxe de l'opérateur ternaire ressemble à ceci :
`condition ? expression1 : expression2`

```
// Without ternary
$isClicked = FALSE;
if ($isClicked) {
  $link_color = "purple";
} else {
  $link_color = "blue";
}
// $link_color = "blue";


// With ternary
$isClicked = FALSE;
$link_color = $isClicked ? "purple"
: "blue";
//  $link_color = "blue";
```

Conditions imbriquées PHP

En PHP, les instructions *conditionnelles imbriquées* approfondissent la complexité des capacités de prise de décision de nos programmes. Ils nous permettent de créer des programmes où chaque décision prise envoie notre programme sur une route différente où il pourrait rencontrer des décisions supplémentaires.

```
$num = 5;

// nested conditional
if ($num > 0){
    echo 'The number is positive. <br>';
      if ($num % 2 === 0){
            echo 'The number is even.';
      }
} else {
      echo 'The number is negative.';
}
```

Opérateurs logiques PHP

En PHP, les expressions qui utilisent des opérateurs logiques évaluent des valeurs booléennes. Les opérateurs logiques incluent :

- ou (`||`)
- et (`&&`)
- exclusif ou (`xor`)
- pas (`!`)

```
// Examples of Logical Operators:

TRUE || TRUE;   // Evaluates to: TRUE
FALSE || TRUE;  // Evaluates to: TRUE
TRUE && TRUE;    // Evaluates to: TRUE
FALSE && TRUE;   // Evaluates to: FALSE
!TRUE;    // Evaluates to: FALSE
!FALSE;   // Evaluates to: TRUE
TRUE xor TRUE;   // Evaluates to: FALSE
FALSE xor TRUE;  // Evaluates to: TRUE
```

PHP && Opérateur

L'opérateur logique `&&` renvoie :

- `TRUE` uniquement si ses deux opérandes ont la valeur true.
- `FALSE` si l'un ou l'autre de ses opérandes est évalué à faux.

```
TRUE && TRUE;    // Evaluates to: TRUE
FALSE && TRUE;   // Evaluates to: FALSE
TRUE && FALSE;   // Evaluates to: FALSE
FALSE && FALSE;  // Evaluates to: FALSE

$passingGrades = TRUE;
$extracurriculars = TRUE;
if ($passingGrades && $extracurriculars){
  echo "You meet the graduation
requirements.";
}
// Prints: You meet the graduation
requirements.
```

PHP ! (pas) Opérateur

En PHP, l'opérateur not (`!`) est utilisé pour inverser une valeur ou une expression booléenne.

```
!TRUE;    // Evaluates to: FALSE
!FALSE;   // Evaluates to: TRUE
```

Priorité des opérateurs PHP

Chaque opérateur en PHP a une *priorité d'opérateur* différente .
Nous pouvons éviter la confusion de la priorité des opérateurs en utilisant des parenthèses pour forcer l'évaluation que nous voulons.

```
TRUE || TRUE && FALSE // Evaluates to: TRUE
(TRUE || TRUE) && FALSE // Evaluates to: FALSE
```

Opérateur PHP Xor

En PHP, l'opérateur logique `xor` signifie ou exclusif.
Il prend deux valeurs ou expressions booléennes différentes comme opérandes et renvoie une seule valeur booléenne.
`xor` prend la valeur `TRUE` **uniquement** si son opérande gauche ou son opérande droit prend la valeur `TRUE` , mais **pas les deux** .

```
TRUE xor TRUE;   // Evaluates to: FALSE
FALSE xor TRUE;  // Evaluates to: TRUE
TRUE xor FALSE;  // Evaluates to: TRUE
FALSE xor FALSE; // Evaluates to: FALSE
```

Opérateurs logiques - Syntaxe alternative

PHP fournit une syntaxe alternative pour l' `||` opérateur — l' `or` opérateur.
Il fournit également une syntaxe alternative pour `&&` l'opérateur — l' `and` opérateur.
Ces opérateurs ont l'avantage de rendre notre code plus lisible par l'homme.

```
// The or Operator:
TRUE or TRUE;   // Evaluates to: TRUE
FALSE or TRUE;  // Evaluates to: TRUE
TRUE or FALSE;  // Evaluates to: TRUE
FALSE or FALSE; // Evaluates to: FALSE

// The and Operator:
TRUE and TRUE;   // Evaluates to: TRUE
FALSE and TRUE;  // Evaluates to: FALSE
TRUE and FALSE;  // Evaluates to: FALSE
FALSE and FALSE; // Evaluates to: FALSE
```

Programmes multi-fichiers : inclure

Un moyen d'améliorer notre code et de séparer les préoccupations est *la modularité* , en séparant un programme en morceaux distincts et gérables où chacun fournit un élément de la fonctionnalité globale. Au lieu d'avoir un programme entier situé dans un seul fichier, le code est organisé en fichiers séparés.
En PHP, les fichiers peuvent être inclus dans un autre fichier avec le mot-clé `include` . Une instruction include est suivie d'une chaîne avec un chemin d'accès au fichier à inclure. Le code du fichier sera exécuté.

```
// one.php
echo "How are";

// two.php
echo " you?";

// index.php
echo "Hello! ";
include "one.php";
include "two.php";
// Prints: Hello! How are you?
```