Introduction au langage PHP

Origine et évolution du PHP

PHP : “Personal Home Page” créé par Rasmus Lerdorf en 1994

En 1997 devient “Hypertext Preprocessor“

Au fur et à mesure ce langage a évolué et des fonctionnalités y sont ajoutées régulièrement.

Aujourd’hui nous sommes à la version 8 de php.

Source : wikipedia

Caractéristiques

- Langage de script côté serveur.
- S'intègre facilement au HTML.
- Prise en charge de diverses bases de données.

Source : Événement Coders Delight

Installation et configuration

- Téléchargez PHP depuis le site officiel et suivez les instructions d'installation.
- Les serveurs locaux comme XAMPP ou WampServer sont également populaires pour faciliter la configuration.
- Configurer le serveur web pour exécuter les scripts PHP.

1er script

Le script "Hello World" ci-dessous affiche une ligne de texte dans le navigateur.

```
<?php
  echo "Hello World!";
?>
```

1er script

Le script "Hello World" ci-dessous affiche une ligne de texte dans le navigateur.

```
<?php
  echo "Hello World!";
?>
```

Éléments de langage

- Balises d’ouverture et de fermeture

Lorsque PHP traite un fichier, il cherche les balises d'ouverture et de fermeture (<?php et ?>) qui délimitent le code qu'il doit interpréter.

- Échappement depuis du HTML

```
<p>Ceci sera ignoré par PHP et affiché au navigateur.</p>
<?php echo 'Alors que ceci sera analysé par PHP.'; ?>
<p>Ceci sera aussi ignoré par PHP et affiché au navigateur.</p>
```

Un fichier avec l’extension .html ne peut pas contenir de php mais un fichier avec l’extension .php peut contenir du html et du php

L'ÉCOLE ATYPIQUE

Variables

Types de variables

1. Entier (`int`) : Représente les nombres entiers.

```
$age = 25;
```

2. Flottant (`float`) : Représente les nombres à virgule flottante.
php

```
$prix = 19.99;
```

3. Chaîne de Caractères (`string`) : Représente du texte.

```
$nom = "John Doe";
```

Une chaîne de caractère peut contenir des chiffres mais on ne pourra pas utiliser cette donnée pour faire des calculs.

Pour écrire un nombre dans une chaîne de caractères on écrira :

```
$codeBarre = “4579896786789”;
```

Un integer n'est pas le même type qu'une string donc :

500 n'est pas la même chose que “500”.

Variables

Types de variables

4. Booléen (`bool`) : Représente les valeurs logiques `true` ou `false`.

php

```
$est_majeur = true;
```

5. Tableau (`array`) : Stocke une collection de valeurs.

```
$fruits = ['pomme', 'banane', 'orange'];
```

6. Objet (`object`) : Instances de classes définies par l'utilisateur.

```
class Utilisateur {

    // Propriétés et méthodes de la classe

}
$utilisateur = new Utilisateur();
```

L'ÉCOLE ATYPIQUE

Variables

Focus sur les tableaux

Syntaxe

2 manières de définir un tableau

- Avec des crochets (nouvelle méthode)

$fruits = [‘orange’, ‘kiwi’, ‘banane’];

- Avec des parenthèses (ancien) :

$fruits = array(‘orange’, ‘kiwi’, ‘banane’);

L'ÉCOLE ATYPIQUE

Variables

Focus sur les tableaux à
1 dimension = liste simple

Un tableau peut avoir 1 ou plusieurs dimensions (=profondeur)

- Exemple de tableau à 1 dimension sans clé :

```
$fruits = ['orange', 'kiwi', 'banane'];
```

- Exemple de tableau à 1 dimension avec clé non définie :

```
$fruits = [
        0 => 'orange',
        1 => 'kiwi',
        2 =>  'banane'
];
```

- Exemple de tableau à 1 dimension avec clé définie :

```
$fruits = [
        'ref1298'  => 'orange',
        'ref9765' => 'kiwi',
        'ref7854' =>  'banane'
];
```

Variables

Focus sur les tableaux à 2 dimensions avec clés

Un tableau peut avoir 1 ou plusieurs dimensions (=profondeur)

- Exemple de tableau à 2 dimensions :

```
$fruits = [
        [
                'nom' => 'orange',
                'prix' => 15,
                'ref' => '5564'
        ],
        [
                'nom' => 'kiwi',
                'prix' => 3,
                'ref' => '5564'
        ],
        [
                'nom' => 'banane',
                'prix' => 5,
                'ref' => '6489'
        ]
];
```

L'ÉCOLE ATYPIQUE

Variables

Déclaration

- **Syntaxe :** Vous pouvez déclarer une variable en utilisant le signe du dollar $, suivi du nom de la variable.

```
$nom_variable = "Valeur de la variable";
```

- **Nom de Variable :** Le nom de la variable doit commencer par un caractère alphabétique ou un trait de soulignement _. Les caractères suivants peuvent être alphanumériques ou des traits de soulignement.

```
$nom = "John";
$_compte_utilisateur = 123;
```

Variables

Déclaration

- **Sensible à la Casse :** PHP est sensible à la casse, ce qui signifie que `$variable` et `$Variable` sont considérées comme deux variables distinctes.

```
$nom = "John";
$Nom = "Doe";
```

- **Assignation de valeur :** Vous pouvez assigner une valeur à la variable au moment de la déclaration ou ultérieurement.

```
$age = 25; // Déclaration avec assignation
$age = 30; // Réassignation de la valeur
```

L'ÉCOLE ATYPIQUE

Variables

Utilisation

- **Modification de Variables :** Vous pouvez modifier la valeur d'une variable en réassignant une nouvelle valeur.

```
$age = 25;

$age = $age + 1; // $age vaut maintenant 26
```

- **Affichage de Variables :** Pour afficher la valeur d'une variable, vous pouvez utiliser la fonction `echo` ou `print`.

```
$prenom = "John";

echo "Le prénom est : " . $prenom; // Affiche "Le prénom est : John"
```

- **Suppression de Variables :** Vous pouvez supprimer une variable en utilisant la fonction `unset`. Cela libère l'espace mémoire associé à la variable.

```
$nom = "Doe";

unset($nom); // Supprime la variable $nom
```

L'ÉCOLE ATYPIQUE

Opérateurs

Arithmétiques

Les opérateurs arithmétiques effectuent des opérations mathématiques de base.

1. **Addition + :** Additionne deux valeurs.

```
$resultat = 5 + 3; // $resultat vaut 8
```

2. **Soustraction - :** Soustrait la deuxième valeur de la première.

```
$resultat = 10 - 4; // $resultat vaut 6
```

3. **Multiplication * :** Multiplie deux valeurs.

```
$resultat = 2 * 3; // $resultat vaut 6
```

4. **Division / :** Divise la première valeur par la deuxième.

```
$resultat = 8 / 2; // $resultat vaut 4
```

5. **Modulo % :** Donne le reste de la division de la première valeur par la deuxième.

```
$reste = 10 % 3; // $reste vaut 1 (car 10 ÷ 3 donne 3 avec un reste de 1)
```

L'ÉCOLE ATYPIQUE

Opérateurs

Assignation

Les opérateurs d'assignation attribuent une valeur à une variable.

- **Assignation = :** Affecte une valeur à une variable.
  `$nombre = 15; // La variable $nombre vaut 15`

- **Opérateurs Composés :** Les opérateurs combinés effectuent une opération et attribuent le résultat à la variable.

```
$total = 5;
$total += 3; // Équivalent à $total = $total + 3; //
$total vaut maintenant 8
```

L'ÉCOLE ATYPIQUE

Opérateurs

Comparaison

Les opérateurs de comparaison comparent deux valeurs et renvoient un résultat booléen.

- **Égalité `==` :** Vérifie si deux valeurs sont égales, sans tenir compte du type.

```
$resultat = (5 == '5'); // $resultat vaut true
```

- **Égalité Stricte `===` :** Vérifie si deux valeurs sont égales en tenant compte du type.

```
$resultat = (5 === '5'); // $resultat vaut false
```

- **Différence `!=` :** Vérifie si deux valeurs sont différentes, sans tenir compte du type.

```
$resultat = (10 != '10'); // $resultat vaut false
```

- **Différence Stricte `!==`** Vérifie si deux valeurs sont différentes en tenant compte du type.

```
$resultat = (10 !== '10'); // $resultat vaut true
```

L'ÉCOLE ATYPIQUE

Opérateurs

Comparaison

Les opérateurs de comparaison comparent deux valeurs et renvoient un résultat booléen.

- `<` : Inférieur à
- `>` : Supérieur à
- `<=` : Inférieur ou égal à
- `>=` : Supérieur ou égal à

Exemples :

```
// Supérieur à
if ($b > $a) {
    echo "$b est supérieur à $a";
} else {
    echo "$b n'est pas supérieur à $a";
}

// Inférieur ou égal à
$c = 5;
if ($c <= $a) {
    echo "$c est inférieur ou égal à $a";
} else {
    echo "$c n'est pas inférieur ou égal à $a";
}
```

L'ÉCOLE ATYPIQUE

Opérateurs

Opérateurs Logiques

Les opérateurs logiques effectuent des opérations logiques sur des expressions booléennes.

- **ET Logique && (et) :** Renvoie true si les deux expressions sont vraies.

```
$resultat = (true && false); // $resultat vaut false
```

- **OU Logique || (ou) :** Renvoie true si au moins l'une des expressions est vraie.

```
$resultat = (true || false); // $resultat vaut true
```

- **NON Logique ! (non) :** Inverse le résultat de l'expression.

```
$resultat = !true; // $resultat vaut false
```

L'ÉCOLE ATYPIQUE

Opérateurs

Incrémentation et de Décrémentation

Les opérateurs d'incrémentation et de décrémentation modifient la valeur d'une variable.

- **Incrémentation ++ :** Ajoute 1 à la valeur de la variable.

```
$nombre = 5;
$nombre++; // Équivalent à $nombre = $nombre + 1; //
$nombre vaut maintenant 6
```

- **Décrémentation -- :** Soustrait 1 à la valeur de la variable.

```
$nombre = 8;
$nombre--; // Équivalent à $nombre = $nombre - 1; //
$nombre vaut maintenant 7
```

Ces opérateurs sont essentiels pour effectuer des calculs, des comparaisons et des opérations logiques dans les scripts PHP. La maîtrise de leur utilisation permet aux développeurs de créer des applications plus complexes et fonctionnelles.

L'ÉCOLE ATYPIQUE

Opérateurs

Chaîne de caractères

Les chaînes de caractères peuvent être concaténées

- `.` : Concaténation de chaînes

```
$str1 = 'Hello';

$str2 = 'World';

$greeting = $str1 . '-' . $str2; //

Retourne "Hello-World"

 . '-' . => ajoute un trait d'union entre les deux chaines
de caractères
```

L'ÉCOLE ATYPIQUE

Structures de Contrôle

Conditions : if

- Permet l'exécution conditionnelle d'une partie de code. (SI)
- IF

Exemple :

```
$age = 34;

if ($age >= 18) {
  echo "Tu es majeur";
} else {
  echo "Tu es mineur";
}
```

Structures de Contrôle

Conditions : else

- Permet l'exécution conditionnelle d'une partie de code. (ALORS)
- ELSE (toujours associé à un IF)

Exemple :

```
$a = 10;

if ($a > 5) {
   echo "La variable a est supérieure à 5.";
} else {
   echo "La variable a n'est pas supérieure à 5.";
}
```

L'ÉCOLE ATYPIQUE

Structures de Contrôle

Boucle : for

- Utilisée pour exécuter un bloc de code un nombre spécifié de fois.

Exemple :

```
for ($i = 1; $i <= 5; $i++) {
    echo $i . " ";
}
// Affiche : 1 2 3 4 5
```

Structures de Contrôle

Boucle : while

- Exécute un bloc de code tant qu'une condition est vraie.

Exemple :

```
$count = 0;

while ($count < 3) {
    echo "Le compteur est à $count. ";
    $count++;
}

// Affiche : Le compteur est à 0. Le compteur est
à 1. Le compteur est à 2.
```

L'ÉCOLE ATYPIQUE

Structures de Contrôle

Boucle : foreach (sans clé)

Tableau à 1 dimension

- Permet de parcourir des tableaux (BOUCLE)

- Permet de répéter des instructions plusieurs fois
- FOREACH

```php
// Définition d'un tableau à 1 dimension

$liste = [‘banane’, ‘orange’, ‘kiwi’];

// Utilisation de la boucle foreach pour parcourir le tableau
foreach ($liste as $fruit) {
    echo $fruit . ‘<br>’;
}

/*
Résultat :
banane
orange
kiwi
*/
```

L'ÉCOLE ATYPIQUE

Structures de Contrôle

Boucle : foreach (sans clé)

Tableau à 2 dimensions

- Permet de parcourir des tableaux (BOUCLE)

- Permet de répéter des instructions plusieurs fois
- FOREACH

```php
// Définition d'un tableau associatif (tableau à 2 dimensions)
$personne = array(
    "nom" => "Doe",
    "prenom" => "John",
    "age" => 30,
    "ville" => "New York"
);

// Utilisation de la boucle foreach pour parcourir le tableau
foreach ($personne as $valeur) {
    echo $valeur . '<br>';
}

/*
Résultat :
nom : Doe
prenom : John
age : 30
ville : New York
*/
```

L'ÉCOLE ATYPIQUE

Structures de Contrôle

Boucle : foreach (avec clé)

Tableau à 2 dimensions

- Permet de parcourir des tableaux (BOUCLE)

- Permet de répéter des instructions plusieurs fois
- FOREACH

```php
// Définition d'un tableau associatif (tableau à 2 dimensions)
$personne = array(
    "nom" => "Doe",
    "prenom" => "John",
    "age" => 30,
    "ville" => "New York"
);

// Utilisation de la boucle foreach pour parcourir le tableau
foreach ($personne as $cle => $valeur) {
    echo "$cle : $valeur <br>";
}

/*
Résultat :
nom : Doe
prenom : John
age : 30
ville : New York
*/
```

L'ÉCOLE ATYPIQUE

Structures de Contrôle

Structure : switch

Permet de tester plusieurs valeurs possibles d'une expression et d'exécuter le code correspondant.

Exemple :

```
$day = "Lundi";

switch ($day) {
    case "Lundi":
        echo "C'est le début de la semaine.";
        break;
    case "Vendredi":
        echo "C'est bientôt le week-end.";
        break;
    default:
        echo "Un autre jour de la semaine.";
}
```

Debug

- print_r($maVariable);

Sert à voir la structure d'un tableau par exemple

Résultat :

Array([0]=>pomme [1]=>banane [2]=>orange)

Exemple d'un print_r d'une chaîne de caractères :
string => 'banane'

L'ÉCOLE ATYPIQUE

Contrôle de qualité

Extension : PHP Stan

PHPStan est un outil d'analyse statique pour PHP, conçu pour détecter des erreurs potentielles dans votre code source sans l'exécuter.

- **Niveaux de vérification :** PHPStan offre plusieurs niveaux de vérification, de 0 à 8, avec un niveau de vérification croissant de la sévérité des erreurs détectées. Un niveau plus élevé détecte un plus grand nombre d'erreurs, mais peut également générer davantage de faux positifs.