Anthony Laplane

PHP Théorie

studi

Introduction

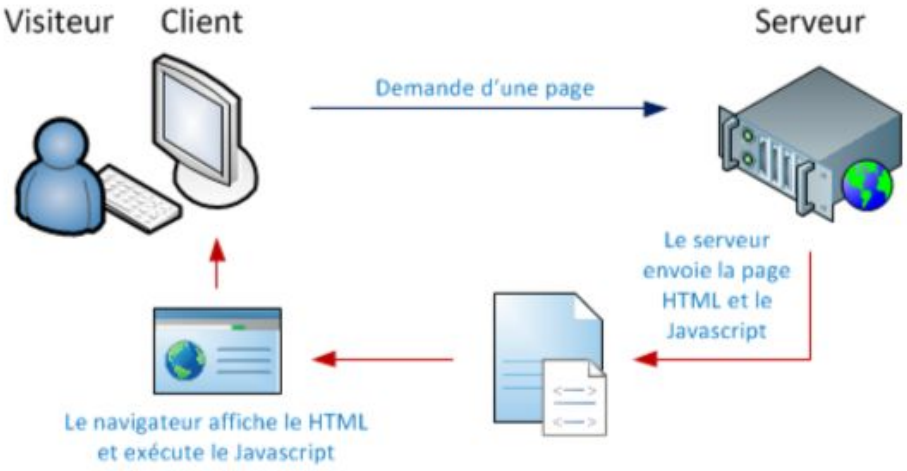
PHP est un langage de programmation tout comme javascript à la différence que javascript est un langage client (lorsqu'il est utilisé sur une page web) alors que php est un langage serveur.

Langage exécuté côté client : Lorsque le code est exécuté, cela se passe sur l'ordinateur de l'utilisateur (le client)

Langage exécuté côté serveur : Lorsque le code est exécuté, cela se passe sur le serveur

Introduction

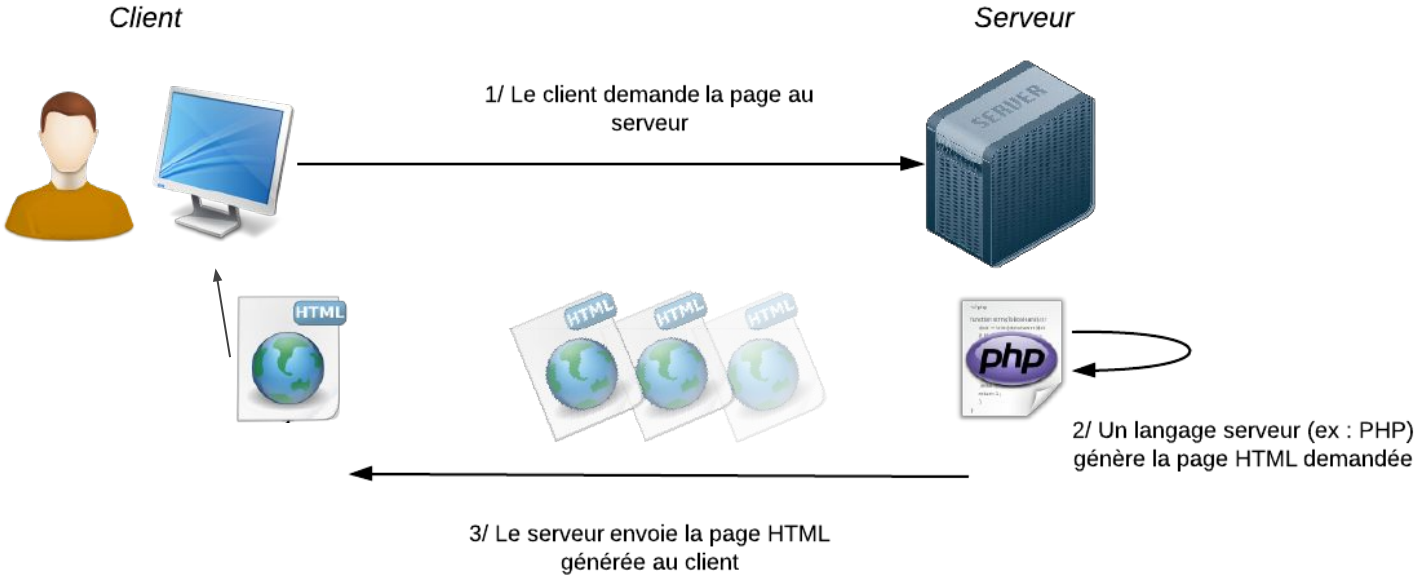
C’est parce que javascript est exécuté côté client qu’il peut modifier la page dans le navigateur (pour ajouter de l'interactivité).



JavaScript est un langage dit client-side, c'est à dire interprété par le client (le navigateur).

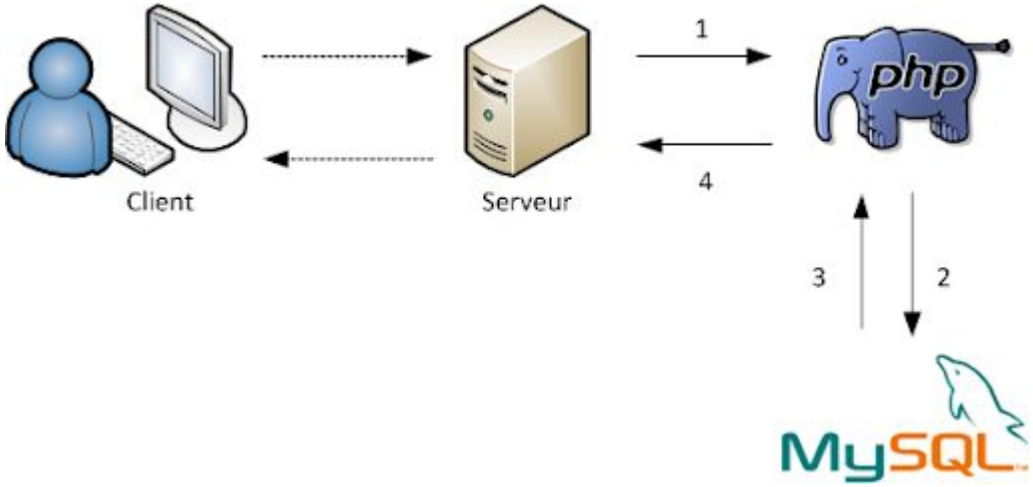
Introduction

PHP va donc être exécuter sur le serveur, avant même que la page soit affichée à l'écran.



Introduction

Pour fonctionner, php a besoin d’un serveur web (apache) et d’une base de données (dans la plupart des cas).



Introduction

Le langage php permet de créer des sites dynamiques avec des données issues d'une base de données (ex: mysql).

La plupart des CMS utilisent le langage PHP (Wordpress, Drupal, Prestashop, Joomla etc.).

Plus de 75% des sites utilisent PHP.

Quelques alternatives à PHP :

Fichier php

Un fichier php se termine par l'extension php (ex: page.php).
Le code php que l'on va écrire dans un fichier php va être entre une balise d'ouverture **<?php** et de fermeture **?>**

```php
<?php

//je peux écrire entre les balises le code php

?>
```

Fichier php

Un fichier php va souvent mélanger du php et du html. Le html doit être écrit en dehors des balises php

```
<h1>Titre en html</h1>

<?php

//je peux écrire entre les balises le code php

?>

<h2>Titre en html</h2>
```

Echo

L’instruction echo va nous permettre d’afficher un message sur la page

```php
<?php
echo 'Bonjour';
// ou
echo('Bonjour');
?>
```

Commentaires

Comme en javascript, on peut ajouter des commentaires dans notre code pour le rendre plus lisible.

```php
<?php
// Commentaire sur une ligne


/*
 Un commentaire qui pourra
 contenir plusieurs ligne
*/
?>
```

Variables

Une variable est une zone mémoire qui pourra contenir une valeur. C’est une sorte de boîte dans laquelle on va pouvoir stocker une information.

```php
<?php
// On déclare une variable age avec comme valeur 16
$age = 16;

// On affiche le contenu de la variable age sur la page avec echo
echo $age;

?>
```

Constantes

En PHP, une constante est une valeur qui ne peut pas être modifiée pendant l'exécution du script. Une constante est définie grâce à la fonction "define()", qui prend deux arguments : le nom de la constante et sa valeur.

```php
<?php

   // Une constante n'est défini qu'une seule fois
   define("_APP_VERSION_", "1.2.6");

   echo _APP_VERSION_;


?>
```

Instruction if

Les conditions en PHP sont définies grâce à des mots-clés tels que "if" (si), "else" (sinon) et "elseif" (sinon si).

```php
<?php
// On déclare une variable age avec comme valeur 16
$age = 16;

// On teste l'âge. Si l'âge est supérieur à 18, on affiche un message, sinon un autre message
if ($age >= 18) {
  echo 'adulte';
} else if ($age >= 13) {
  echo 'ado';
} else {
  echo 'enfant';
}
?>
```

Echo suite

L'instruction echo va nous permettre d'afficher un message mais aussi des variables avec guillet simple ou double.
La différence étant qu'avec guillemet double, php affichera le contenu de la variable alors que ce n'est pas le cas avec guillemet simple :

```php
<?php
$age = 30;

echo "age : $age ans"; // Affichage -> age : 30 ans
echo 'age : $age ans'; // Affichage -> age : $age ans

// Un raccourci pour echo intéressante quand on mélange avec du html :
?>
<p>Vous avez <?=$age; ?> ans</p>
```

Instruction switch

switch permet de comparer une expression à une liste de valeurs. Elle peut être utilisée de manière similaire à la structure "if"/"elseif"/"else", mais peut être plus concise et plus lisible dans certains cas.

```php
<?php
$numJour = 3;

switch ($numJour) {
    case 1:
        echo "Lundi";
        break;
    case 2:
        echo "Mardi";
        break;
    case 3:
        echo "Mercredi";
        break;
    default:
        echo "Numéro de jour invalide";
}
?>
```

Les opérateurs

- Opérateurs de comparaison de valeur :
  - ">" (strictement supérieur à)
  - "<" (strictement inférieur à)
  - ">=" (supérieur ou égal à)
  - "<=" (inférieur ou égal à)
  - "==" (égal à) ou "===" (pour vérifier le type)
  - "!=" (différent de) ou "!==" (pour vérifier le type)
- Opérateurs logiques en PHP :
  - "&&" (ET logique)
  - "||" (OU logique)
- Opérateurs arithmétiques :
  - "+" (addition)
  - "-" (soustraction)
  - "*" (multiplication)
  - "/" (division)
  - "%" (modulo)

Les tableaux

Un tableau (array en anglais) est une structure de données qui permet de stocker une liste d'éléments de manière organisée. Un tableau peut contenir des éléments de n'importe quel type (chaînes de caractères, nombres, objets, etc.).

```php
<?php
    // Tableau simple
    $fruits = ["Banane", "Orange", "Pomme"];
    echo $fruits[0];  // Affiche "Banane"

    // Tableau associatif
    $utilisateur =["nom" => "Dupond", "prenom" => "Jean", "age" => 26];
    echo $utilisateur["nom"];  // Affiche "Dupond"
?>
```

Note : Il existe un syntaxe alternative :

```php
$fruits = array("Banane", "Orange", "Pomme");
```

Les tableaux multidimensionnels

un tableau multidimensionnel est un tableau qui contient d'autres tableaux en tant qu'éléments. Cela permet de créer des structures de données plus complexes pour stocker et manipuler des informations.

```php
<?php
$utilisateurs = [
    ["nom" => "Dupond", "prenom" => "Jean", "age" => 26],
    ["nom" => "Martin", "prenom" => "Rose", "age" => 35],
    ["nom" => "Doe", "prenom" => "Jane", "age" => 31]
];

?>
```

Boucle foreach

La boucle "foreach" permet de parcourir un tableau et d'exécuter un bloc de code pour chaque élément du tableau. La boucle "foreach" prend en compte les clés et les valeurs du tableau.

Exemple sans clé

```php
<?php
    $fruits =["Banane", "Orange", "Pomme"];

    foreach ($fruits as $fruit) {
        echo "<p>$fruit</p>";
    }
?>
```

Boucle foreach

Exemple avec clé

```php
<?php
    $fruits = ["Banane" => "Jaune", "Orange" => "Orange", "Pomme" => "Verte"];

    foreach ($fruits as $fruit => $couleur) {
        echo "<p>$fruit est de couleur $couleur</p>";
    }
?>
```

Boucle foreach

Exemple avec tableau multidimensionnel

```php
<?php
$utilisateurs = [
    ["nom" => "Dupond", "prenom" => "Jean", "age" => 26],
    ["nom" => "Martin", "prenom" => "Rose", "age" => 35],
    ["nom" => "Doe", "prenom" => "Jane", "age" => 31]
];

foreach ($utilisateurs as $key=>$utilisateur) {
    echo "<h2>Nom : " . $utilisateur["nom"] . "</h2>";
    echo "<p>Prénom : " . $utilisateur["prenom"] . "</p>";
    echo "<p>Age : " . $utilisateur["age"] . "</p>";
}
?>
```

Boucle for

La boucle "for" permet d'exécuter un bloc de code plusieurs fois de manière itérative. La boucle "for" se compose de trois parties : l'initialisation, la condition de fin de boucle et l'incrémentation.

```php
<?php
    for ($i = 1; $i <= 10; $i++) {
        echo "$i ";
    }
    // Affiche : 1 2 3 4 5 6 7 8 9 10
?>
```

Boucle while

La boucle "while" permet d'exécuter un bloc de code de manière itérative tant qu'une condition est vraie. Elle se compose d'une condition qui est évaluée avant chaque itération de la boucle.

Exemple sans clé

```php
<?php
    $i = 1;
    while ($i <= 10) {
        echo "$i ";
        $i++;
    }
    // Affiche : 1 2 3 4 5 6 7 8 9 10
?>
```

Include et require

Les instructions "include" et "require" permettent d'inclure et d'exécuter un fichier au sein d'un autre fichier. Elles permettent de partager du code commun entre plusieurs fichiers et de rendre celui-ci plus lisible et maintenable.

```php
<?php
include 'fichier1.php';
require 'fichier2.php';
?>
```

La différence entre ces deux instructions est que require va générer une exception si le fichier est introuvable alors qu’include affichera seulement un warning.

Include_once et require_once

"include_once" et "require_once" fonctionnent de manière similaire à "include" et "require", mais avec une différence importante : si le fichier inclus a déjà été inclus dans le script en cours d'exécution, il ne sera pas inclus de nouveau.

```php
<?php
include_once 'fichier1.php';
require_once 'fichier2.php';
?>
```

$_GET

$_GET est un tableau associatif qui contient les données envoyées au serveur via la méthode "GET" d'un formulaire HTML ou d'une URL.

Les données passées par une adresse url sont de la forme suivante :

mapage.php**?**nom**=**dupond**&**prenom**=**jean

On pourra ensuite y accéder en php de la manière suivante :

```php
<?php
echo $_GET['prenom'];
if (isset($_GET['nom'])) {
    echo $_GET['nom'];
}
?>
```

$_POST

$_POST est un tableau associatif qui contient les données envoyées au serveur via la méthode "POST" d'un formulaire HTML.

On pourra ensuite y accéder en php de la manière suivante:

```php
<?php
echo $_POST['prenom'];
if (isset($_POST['nom'])) {
    echo $_POST['nom'];
}
?>
```

Les fonctions

Une fonction est un bloc de code qui peut être exécuté de manière autonome et qui peut éventuellement retourner une valeur. Une fonction peut être appelée à plusieurs reprises dans un programme, ce qui permet de réutiliser du code et de rendre celui-ci plus lisible et maintenable.

```php
<?php
function carre($nombre) {
    return $nombre * $nombre;
}

echo carre(5);  // Affiche 25

?>
```

PDO

PDO (PHP Data Objects) est une extension de PHP qui permet de gérer les bases de données de manière uniforme, quel que soit le type de base de données utilisé (MySQL, SQLite, etc.). Il permet de se connecter à une base de données, d'exécuter des requêtes SQL et de récupérer les résultats. Il offre également des fonctionnalités de sécurité pour protéger contre les injections SQL.

PDO

Ci-dessous un exemple d'une requête **NON SÉCURISÉ**

```php
<?php
$pdo = new PDO('mysql:dbname=my_db;host=localhost;charset=utf8mb4', 'root', '');
$id = $_GET['id'];
$query = $pdo->prepare("SELECT * FROM users WHERE id = $id");
$query->execute();
$result = $query->fetch(PDO::FETCH_ASSOC);
?>
```

Ce code n'est pas sécurisé car un attaquant pourra injecter du code dans le paramètre d'url :

?id=5;DELETE FROM users;

La requête suivante sera alors exécutée :

SELECT * FROM users WHERE id =5;DELETE FROM users;

PDO

PDO nous permet de préparer, exécuter et récupérer des données.

On commence par instancier un objet PDO dans un try/catch pour gérer les erreurs.

```php
<?php
try
{
    $pdo = new PDO("mysql:dbname=my_db;host=localhost;charset=utf8mb4", "root", "");
}
catch (Exception $e)
{
        die('Erreur : ' . $e->getMessage());
}
?>
```

PDO

Avec l'objet PDO nous pouvons effectuer des requêtes préparées

La préparation de requête avec bindParam() nous permet de sécuriser nos requêtes.

```php
<?php
$id = (int)$_GET['id'];
$query = $pdo->prepare("SELECT * FROM users WHERE id = :id");
$query->bindValue(':id', $id, PDO::PARAM_INT);
$query->execute();
$result = $query->fetch(PDO::FETCH_ASSOC);
?>
```