Introduction au développement côté serveur

I. Quelle est la différence entre côté client et côté serveur ?

Côté Client :

- Le côté client fait référence à la partie d'une application web qui s'exécute sur le navigateur ou le périphérique de l'utilisateur.
- Les technologies côté client comprennent HTML, CSS et JavaScript.
- Les interactions utilisateur, l'affichage et certaines logiques sont gérées côté client.

Côté Serveur :

- Le côté serveur concerne la partie d'une application qui s'exécute sur le serveur.
- Il gère les logiques métier, l'accès à la base de données, et génère souvent le contenu dynamique.
- Les technologies côté serveur incluent PHP, Node.js, Python (Django), etc.

II. Quelle est la relation entre client et serveur ?

Les applications web fonctionnent en utilisant une architecture client-serveur.

Le client envoie des requêtes au serveur, et le serveur répond avec des données ou exécute des actions.

Cette communication est généralement basée sur le protocole HTTP/HTTPS.

- **Requête HTTP côté client (utilisant Fetch en JavaScript) :**

```
// Envoi d'une requête GET au serveur
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Erreur :', error));
```

- **Réponse HTTP côté serveur (Node.js) :**

```
// Serveur répondant à une requête GET
const http = require('http');
const server = http.createServer((req, res) => {
  res.writeHead(200, {'Content-Type': 'application/json'});
  res.end(JSON.stringify({ message: 'Données récupérées avec succès' }
});

server.listen(3000, () => {
  console.log('Serveur en cours d\'écoute sur le port 3000');
});
```

III. Quelle est la différence entre HTTP et HTTPS ?

HTTP (Hypertext Transfer Protocol) :

- Protocole de communication utilisé pour transférer des données sur le web.
- Les requêtes sont généralement de type GET (récupérer des données) ou POST (envoyer des données).

HTTPS (Hypertext Transfer Protocol Secure) :

- Version sécurisée de HTTP utilisant une couche de chiffrement SSL/TLS.
- Assure la confidentialité et l'intégrité des données échangées entre le client et le serveur.

Quelques raisons de choisir HTTPS :

- **Sécurité des données** : HTTPS chiffre les données échangées entre le client et le serveur, rendant plus difficile pour les tiers non autorisés d'intercepter et de comprendre ces données. Protège contre les attaques de type "man-in-the-middle" où un attaquant tente d'intercepter ou de modifier les données pendant la transmission.
- **Confiance des utilisateurs** : La présence d'un certificat SSL/TLS et le cadenas dans la barre d'adresse du navigateur renforcent la confiance des utilisateurs. Les utilisateurs sont plus enclins à partager des informations sensibles (comme les informations de carte de crédit) sur des sites HTTPS.
- **Intégrité des données** : HTTPS garantit l'intégrité des données pendant la transmission. Si des données sont modifiées pendant le transfert, le navigateur avertira l'utilisateur.

- **Protection contre la falsification de sites Web** : Les certificats SSL/TLS utilisés dans HTTPS garantissent l'authenticité du site web. Cela aide à prévenir les attaques de type "phishing" où des sites malveillants tentent de se faire passer pour des sites légitimes.

- **Amélioration du classement dans les moteurs de recherche** : Les moteurs de recherche, tels que Google, privilégient les sites sécurisés par HTTPS dans leurs résultats de recherche. Avoir HTTPS peut donc améliorer le référencement de votre site web.

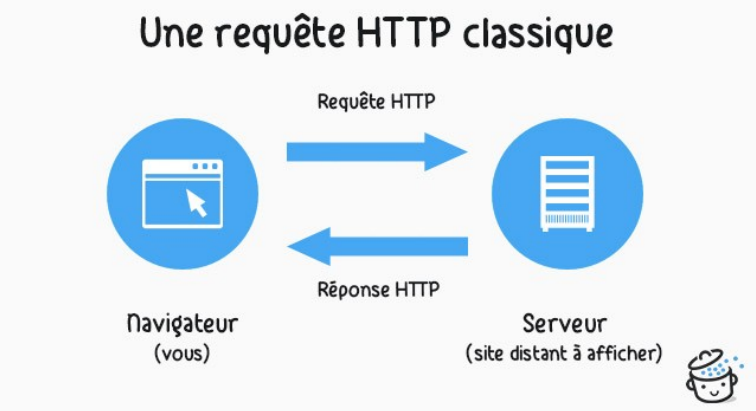
- **Respect des réglementations de confidentialité** : Certains règlements, tels que le Règlement général sur la protection des données (RGPD) en Europe, exigent la protection des données personnelles. L'utilisation de HTTPS contribue à respecter ces réglementations.

- **Compatibilité avec les nouvelles fonctionnalités du navigateur** : De nombreuses fonctionnalités modernes du navigateur, telles que les Service Workers (pour les applications web progressives) et les API pour les appareils, sont disponibles uniquement sur des sites sécurisés.

- **Protection contre les injections de code** : HTTPS offre une protection supplémentaire contre les attaques de type "injection de code", où des attaquants insèrent du code malveillant dans les communications entre le client et le serveur.

- **Conformité aux normes de sécurité** : Utiliser HTTPS est devenu une meilleure pratique et une norme de sécurité pour les sites web. Cela montre que vous prenez au sérieux la sécurité en ligne.

IV. Quelles sont les différents types de requêtes / réponses ? ?



Requêtes HTTP :

- **GET** : Utilisé pour récupérer des données du serveur.

- **POST** : Utilisé pour envoyer des données au serveur.

- **PUT, DELETE, etc.** : Utilisés pour d'autres opérations spécifiques.

Réponses HTTP :

- 1xx (Informationnel) : Informations.
- 2xx (Succès) : La requête a été acceptée avec succès.

  ➔ 200 OK : La requête a réussi.

  ➔ 201 Created : La requête a été correctement traitée et une nouvelle ressource a été créée.
- 3xx (Redirection) : Le client doit effectuer une action supplémentaire pour compléter la requête.

  ➔ 301 Moved Permanently : La ressource a été déplacée de manière permanente.

  ➔ 302 Found : La ressource a été trouvée, mais elle se trouve temporairement à une autre URL.
- 4xx (Erreur client) : La requête contient une mauvaise syntaxe ou ne peut pas être accomplie par le serveur.

  ➔ 400 Bad Request : La requête est mal formulée.

  ➔ 404 Not Found : La ressource demandée n'a pas été trouvée.
- 5xx (Erreur serveur) : Le serveur a échoué à accomplir une requête valable.

  ➔ 500 Internal Server Error : Erreur interne du serveur.

4. Quelle est la différence entre un site statique et un site dynamique ?

SSG - Static-Site-Generation = Site Statique : Page HTML simple sans interaction côté serveur.

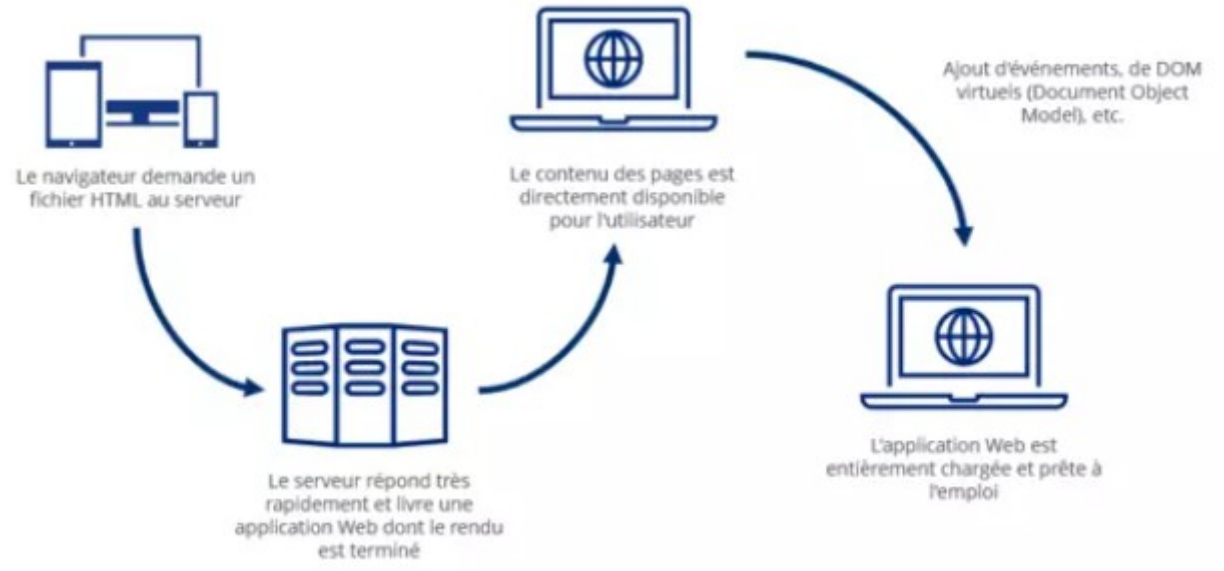
a) **Contenu Fixe** : Un site statique a un contenu fixe qui ne change pas fréquemment. Le contenu est préalablement créé et stocké sous forme de fichiers (HTML, CSS, JavaScript, images).

b) **Génération Côté Développeur** : Le développeur génère les pages du site lors de la phase de développement. Chaque page est une entité distincte, et il n'y a généralement pas de base de données sous-jacente.

c) **Avantages** : Facile à déployer car il suffit de télécharger les fichiers sur un serveur web. Performant en termes de vitesse d'accès car les fichiers sont statiques et peuvent être mis en cache.

d) **Inconvénients** :Limité en termes d'interactivité et de personnalisation. Gestion complexe pour les sites avec beaucoup de pages à maintenir. Non adapté si changements réguliers.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1
    <title>Page Statique</title>
</head>
<body>
    <h1>Bienvenue sur notre site statique !</h1>
    <p>Ce contenu ne change pas fréquemment.</p>
</body>
</html>
```



CSR - Client-Side-Rendering / SSR – Server Side Rendering = Site Dynamique : Page générée côté serveur ou côté client, à la demande.

a) **Contenu Généré à la Demande** : Un site dynamique génère le contenu à la demande en fonction des actions de l'utilisateur ou d'autres facteurs. Les pages ne sont pas préalablement créées mais sont générées au moment de la requête.

b) **Base de Données** : Les sites dynamiques utilisent souvent une base de données pour stocker et récupérer des données. Les informations sont stockées de manière dynamique et peuvent être modifiées en temps réel.

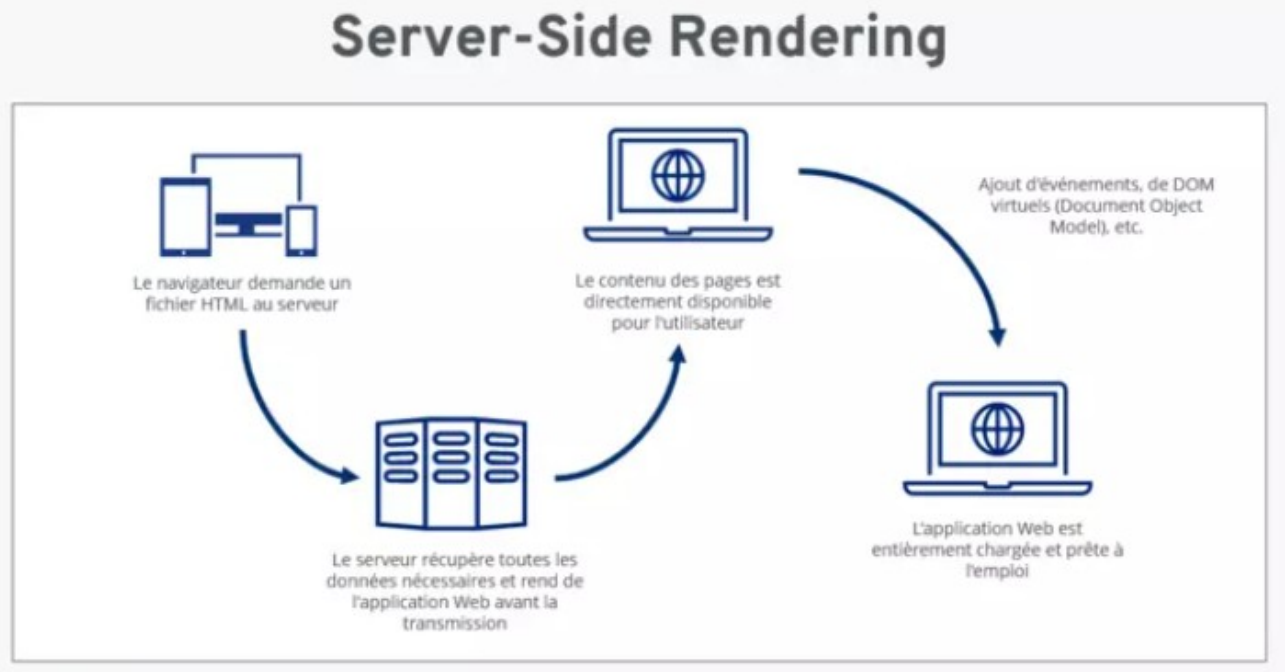
c) **Interactivité** : Offre une interactivité élevée. Les utilisateurs peuvent interagir avec le site, par exemple en remplissant des formulaires, en laissant des commentaires, etc.

d) **Avantages** : Facilité de gestion du contenu, en particulier pour les sites avec beaucoup de pages ou des mises à jour fréquentes. Possibilité d'offrir une expérience utilisateur personnalisée.

e) **Inconvénients** : Requiert généralement une configuration de serveur plus complexe. Peut être moins performant en termes de temps de chargement par rapport aux sites statiques.

f) **Exemple** : Un site e-commerce qui affiche des produits en fonction des choix de l'utilisateur.

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('<h1>Bienvenue sur notre site dynamique !</h1
```



Avantages du Server-Side Rendering (SSR) :

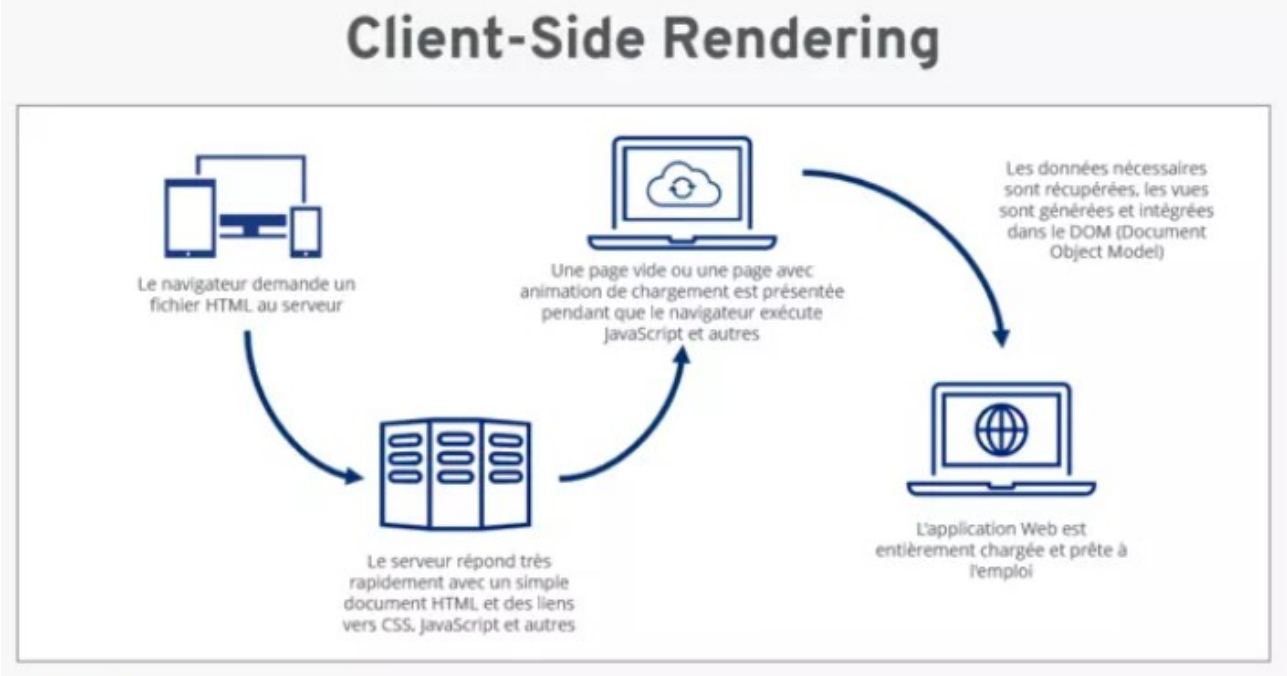
- Préchargement des pages par le serveur, offrant une expérience utilisateur rapide.
- Optimisation de la vitesse de chargement, surtout pour les sites statiques.
- Impact positif sur le référencement grâce à une meilleure indexation par les moteurs de recherche.

Inconvénients du Server-Side Rendering (SSR) :

- Sollicitation élevée de la capacité du serveur.
- Moins adapté aux sites nécessitant de nombreuses interactions utilisateur ou de nombreuses demandes fréquentes.
- Risque d'annulation des avantages de chargement rapide dans le cas de projets avec une forte demande sur le serveur.

En résumé, le SSR offre une expérience utilisateur rapide avec un chargement optimisé, mais peut être limité par la capacité du serveur, particulièrement pour les sites nécessitant beaucoup d'interactions ou de demandes fréquentes.

Les langages de programmation pour le server-side rendering peuvent être les suivants : Java, Ruby, ASP.NET, Perl, PHP, Python, Node.js ou JavaScript



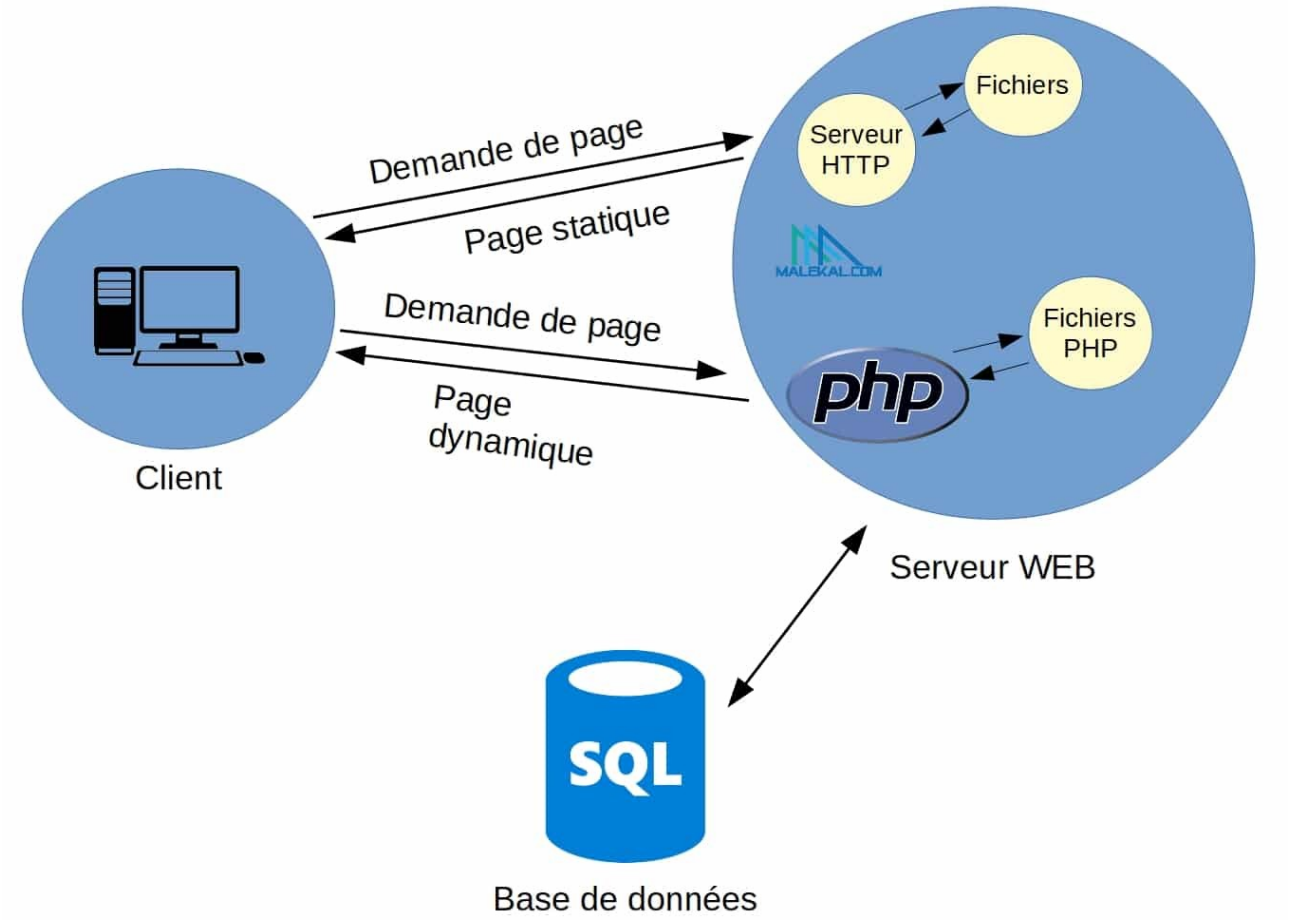
Avantages du Client-Side Rendering (CSR) :

- Chargement initial du site plus rapide, rendu des pages suivantes rapide.
- Meilleure expérience utilisateur avec des interactions fluides.
- Possibilité d'optimiser le rendu en ne rechargeant que les éléments nécessaires.
- Visibilité du code source pour les utilisateurs.

Inconvénients du Client-Side Rendering (CSR) :

- Problèmes d'indexation pour les moteurs de recherche.
- Dépendance à JavaScript, impactant les extensions de navigateur et les temps de chargement.
- Risques potentiels pour le SEO en raison de difficultés d'indexation.

En résumé, le CSR offre une expérience utilisateur améliorée avec des chargements rapides, mais présente des défis liés à l'indexation pour les moteurs de recherche et à la dépendance à JavaScript.



SSR vs CSR vs SSG : pour résumer

Le server-side rendering offre une excellente vitesse de **chargement de page**, qui est en revanche associée à la **sollicitation élevée du serveur Web**. Le client-side rendering fonctionne inversement et ménage le serveur en affichant la majeure partie de la page dans le navigateur, à condition que l'utilisateur n'ait pas bloqué JavaScript. La génération de site statique protège à la fois le serveur et le client et garantit une **livraison rapide du contenu** à travers l'approche de prérendu, à moins qu'il ne s'agisse d'un contenu interactif et soumis à des changements constants.