**JavaScript**

* Questions

1. Expliquez la différence entre let, const et var en JavaScript.
2. Comment fonctionne la boucle d'événements (event loop) en JavaScript ?
3. Qu'est-ce que les promesses (Promises) en JavaScript et comment gèrent-elles les opérations asynchrones ?
4. Pouvez-vous expliquer ce qu'est une closure en JavaScript ?
5. Qu'est-ce que le JSON et comment JavaScript l'utilise-t-il pour gérer les données ?
6. Expliquez la différence entre les fonctions synchrones et asynchrones en JavaScript.
7. Comment les arrow functions en JavaScript diffèrent-elles des fonctions traditionnelles ?
8. Qu'est-ce que le DOM en JavaScript et comment interagit-il avec une page web ?
9. Comment gérer les erreurs en JavaScript avec try/catch ?

* Réponses

1. let, const et var

let : portée de bloc, la variable est déclarée à l'endroit où elle est utilisée pour la première fois

const : constante, la variable ne peut pas être redéfinie

var : portée de fonction, la variable est déclarée au début de la fonction

2. Boucle d'événements

La boucle d'événements est un mécanisme qui permet à JavaScript de traiter les événements asynchrones, tels que les requêtes HTTP ou les événements utilisateur. La boucle d'événements est une boucle infinie qui exécute les fonctions liées aux événements qui sont prêts à être traités.

3. Promesses

Les promesses sont un mécanisme asynchrone qui permettent de gérer les opérations asynchrones de manière non bloquante. Une promesse est un objet qui représente une valeur ou une erreur qui sera disponible à l'avenir.

4. Closure

Une closure est une fonction qui capture l'environnement local dans lequel elle a été définie. Cela signifie que la fonction peut accéder aux variables et fonctions locales de son environnement, même après que la fonction ait été exécutée.

5. JSON

JSON est un format de données texte qui est couramment utilisé pour échanger des données entre des applications. JSON est un format léger et facile à lire, ce qui le rend idéal pour l'échange de données entre des applications différentes.

6. Fonctions synchrones et asynchrones

Les fonctions synchrones sont exécutées immédiatement, tandis que les fonctions asynchrones sont exécutées plus tard. Les fonctions asynchrones sont utiles pour les opérations qui ne peuvent pas être exécutées immédiatement, telles que les requêtes HTTP ou les événements utilisateur.

7. Arrow functions

Les arrow functions sont une nouvelle syntaxe de fonction introduite dans ECMAScript 6. Les arrow functions sont plus concises que les fonctions traditionnelles et ont un comportement légèrement différent.

8. DOM

Le DOM, ou Document Object Model, est une représentation arborescente d'une page web. Le DOM permet aux applications JavaScript d'interagir avec les éléments d'une page web.

9. try/catch

La syntaxe try/catch permet de capturer les erreurs dans le code JavaScript. La syntaxe try/catch permet d'exécuter un bloc de code et de capturer les erreurs qui se produisent dans ce bloc.

**React**

* Questions

1. Quels sont les composants en React et comment diffèrent-ils des fonctions JavaScript traditionnelles ?
2. Comment React améliore-t-il l'expérience utilisateur dans les applications web ? Pouvez-vous donner un exemple d'utilisation des hooks dans React ?
3. Expliquez le concept d'état (state) en React. Comment est-il utilisé ?
4. Décrivez le cycle de vie d'un composant React. Quelles sont certaines des méthodes de cycle de vie ?
5. Qu'est-ce que les props en React et comment sont-elles utilisées ?
6. Expliquez la différence entre un composant de classe et un composant fonctionnel en React.
7. Qu'est-ce que le JSX et comment est-il utilisé en React ?
8. Comment pouvez-vous gérer les formulaires en React ?
9. Qu'est-ce que le lifting state up en React et pourquoi est-il important ?
10. Comment implémenteriez-vous le routage dans une application React ?
11. Quels sont les hooks en React et pouvez-vous donner un exemple d'utilisation ?

* Réponses

1. Composants

Fonctions JavaScript avec des propriétés et un état qui represente une partie d’interface utilisateur

Plus organisés et maintenables

2. Expérience utilisateur

Mise à jour de l'interface utilisateur de manière réactive

Améliore les performances

3. État

Données qui changent au fil du temps

Utilisé pour gérer l'interface utilisateur

4. Cycle de vie

Série d'événements qui se produisent au cours de la vie d'un composant

Permet de contrôler le comportement du composant

5. Props

Données transmises à un composant

Utilisées pour personnaliser le comportement du composant

6. Composants de classe

Utilisent des classes et des méthodes

Plus traditionnels

7. JSX

Syntaxe qui permet d'insérer du code JavaScript dans du HTML

Rend le développement React plus simple

8. Formulaires

Utilisent des composants React pour créer des formulaires

Permet de gérer les données de formulaire de manière réactive

9. Lifting state up

Transfert de l'état d'un composant enfant à un composant parent

Permet de gérer l'état de manière centralisée

10. Routage

Permet aux utilisateurs de naviguer entre différentes pages d'une application

Utilise des composants React pour créer des routes

11. Hooks

Fonctions qui permettent d'utiliser l'état et les props dans les composants fonctionnels

Plus simples et plus concises que les classes

Exemples d'utilisation des hooks

useState() pour gérer l'état

useEffect() pour exécuter du code lorsque l'état ou les props changent

useContext() pour accéder à des données contextuelles

**Express.js**

* Questions

1. Qu'est-ce qu'Express.js et pourquoi est-il fréquemment utilisé dans les applications Node.js ?
2. Quels sont les avantages d'utiliser Express.js dans le développement de serveurs Node.js ?
3. Comment gérez-vous le routage dans une application Express.js ?
4. Expliquez le concept de middleware dans Express.js.
5. Comment pouvez-vous sécuriser une application Express.js ?
6. Qu'est-ce que CORS et comment le gérez-vous dans Express.js ?
7. Comment géreriez-vous les sessions utilisateur dans Express.js ?
8. Expliquez comment les données sont envoyées et reçues dans une API Express.js.
9. Comment intégreriez-vous une base de données MongoDB avec Express.js ?
10. Quels sont les avantages de l'utilisation d'Express.js pour le développement de serveur en Node.js ?
11. Comment gérez-vous les erreurs dans une application Express.js ?

* Réponses

1. Express.js

Framework web léger pour Node.js

Simplifie le développement de serveurs web

2. Avantages d'Express.js

Simple à apprendre et à utiliser

Extensif et personnalisable

Performant

3. Routage

Utilisation de routes pour gérer les demandes HTTP

Permet de créer des applications web dynamiques

4. Middleware

Fonctions qui sont exécutées avant ou après le traitement d'une demande

Permet d'ajouter des fonctionnalités supplémentaires aux applications

5. Sécurité

Utilisation de méthodes de sécurité telles que l'authentification et l'autorisation

Protection des données sensibles

6. CORS

Permet aux applications web d'accéder aux ressources d'autres domaines

Régule les demandes inter-domaines

7. Sessions utilisateur

Utilisation de cookies ou de sessions pour suivre les utilisateurs

Permet de fournir une expérience utilisateur personnalisée

8. Données

Envoie et réception de données en utilisant des méthodes HTTP telles que GET, POST, PUT et DELETE

9. MongoDB

Utilisation du module mongoose pour connecter Express.js à MongoDB

Permet de stocker des données dans une base de données NoSQL

10. Avantages

Rapide, simple et extensible

Large communauté et documentation

11. Erreurs

Utilisation de méthodes telles que catch() et next() pour gérer les erreurs

Permet d'afficher des messages d'erreur aux utilisateurs

**Node.js**

* Questions

1. Qu'est-ce que Node.js et en quoi diffère-t-il du JavaScript côté client ?
2. Quels sont les avantages de Node.js pour le développement côté serveur ? Pouvez-vous me donner un exemple d'utilisation de Node.js dans une application web ?
3. Comment Node.js gère-t-il les opérations asynchrones ?
4. Quelles sont les utilisations courantes de Node.js ?
5. Expliquez le rôle du module http en Node.js.
6. Comment utilisez-vous les modules en Node.js ?
7. Qu'est-ce que NPM et comment l'utilisez-vous en Node.js ?
8. Comment Node.js traite-t-il les requêtes simultanées ?
9. Quelles sont les différences entre Node.js et d'autres serveurs backend comme PHP ou Java ?
10. Expliquez ce que sont les Stream en Node.js et donnez un exemple de leur utilisation.
11. Comment gérez-vous les dépendances de projet en Node.js ?

* Réponses

1. Node.js

Runtime JavaScript côté serveur

Permet d'utiliser JavaScript pour créer des applications web, des API et des microservices

2. Avantages de Node.js

Performant

Simple à apprendre et à utiliser

Multiplateforme

3. Opérations asynchrones

Node.js utilise un modèle d'E/S asynchrone

Les opérations asynchrones permettent à Node.js de traiter plusieurs requêtes simultanément

4. Utilisations courantes

Serveurs web

API

Microservices

Streaming

5. Module http

Fournit des fonctions pour créer et gérer des serveurs HTTP

6. Modules

Les modules sont des fichiers JavaScript qui contiennent du code réutilisable

Ils sont chargés dynamiquement par Node.js

7. NPM

Gestionnaire de packages Node.js

Permet d'installer et de gérer des modules Node.js

8. Requêtes simultanées

Node.js utilise un modèle d'E/S asynchrone pour traiter plusieurs requêtes simultanément

9. Différences avec d'autres serveurs backend

Node.js est un langage événementiel

D'autres serveurs backend sont souvent basés sur des threads

10. Streams

Les streams sont des flux de données continus

Ils peuvent être utilisés pour transférer des données entre des applications ou des composants

11. Dépendances de projet

Les dépendances de projet sont des modules Node.js utilisés par un projet

Elles sont gérées par NPM

**MongoDB**

* Questions

1. Qu'est-ce que MongoDB et en quoi diffère-t-il des bases de données relationnelles ?
2. Comment effectuez-vous les opérations CRUD de base (Créer, Lire, Mettre à jour, Supprimer) dans MongoDB ?
3. Expliquez l'indexation dans MongoDB et pourquoi elle est importante.
4. Qu'est-ce qu'un document en MongoDB et comment est-il structuré ?
5. Comment gérez-vous les relations entre les données dans MongoDB ?
6. Quels sont les avantages d'utiliser MongoDB dans un stack MERN ?
7. \*\*Comment utilisez-vous les agrégations en MongoDB ?

* Réponses

43. Qu'est-ce que MongoDB et en quoi diffère-t-il des bases de données relationnelles ?

Base de données NoSQL orientée documents

Pas de schéma prédéfini

Plus flexible et scalable

44. Comment effectuez-vous les opérations CRUD de base (Créer, Lire, Mettre à jour, Supprimer) dans MongoDB ?

db.collection.insert()

db.collection.find()

db.collection.update()

db.collection.delete()

45. Expliquez l'indexation dans MongoDB et pourquoi elle est importante.

Permet de rechercher des données plus rapidement

Utilise des index uniques et non uniques

46. Qu'est-ce qu'un document en MongoDB et comment est-il structuré ?

Objet JSON

Possède des champs et des valeurs

47. Comment gérez-vous les relations entre les données dans MongoDB ?

Utilisez des collections liées

Utilisez des clés étrangères

48. Quels sont les avantages d'utiliser MongoDB dans un stack MERN ?

Facile à intégrer avec React et Express

Performant et scalable

Flexible et évolutif

49. Comment utilisez-vous les agrégations en MongoDB ?

Pour regrouper et analyser les données

Utilisez des opérateurs d'agrégation tels que group(), match() et sort()

**Questions sur le Développement Web:**

* Questions

"Comment assurez-vous la responsivité de vos applications web ?"

"Pouvez-vous expliquer le concept de l'API RESTful et son importance dans le développement web moderne ?"

* Réponses

Comment assurer la responsivité de vos applications web ?

Il existe de nombreuses façons d'assurer la responsivité de vos applications web. Voici quelques conseils :

Optimisez votre code. Assurez-vous que votre code est bien écrit et optimisé pour les performances. Cela signifie d'utiliser des structures de données efficaces, d'éviter les boucles inutiles et d'utiliser des fonctions de bibliothèque performantes.

Utilisez des ressources en cache. Le cache est un excellent moyen d'améliorer les performances de votre application en stockant les données fréquemment utilisées en mémoire.

Utilisez une architecture distribuée. Si votre application nécessite de traiter un grand volume de trafic, vous pouvez utiliser une architecture distribuée pour répartir la charge sur plusieurs serveurs.

Utilisez des techniques de mise en cache en amont. Si votre application doit accéder à des ressources externes, vous pouvez utiliser des techniques de mise en cache en amont pour stocker ces ressources localement.

Voici quelques exemples spécifiques de techniques que vous pouvez utiliser pour améliorer la responsivité de vos applications web :

Utilisez des images optimisées. Les images sont une source majeure de lenteur dans les applications web. Vous pouvez réduire la taille de vos images en utilisant des outils tels que ImageOptim ou TinyPNG.

Utilisez des CDN. Les CDN (Content Delivery Networks) sont des réseaux de serveurs répartis dans le monde entier. Ils peuvent être utilisés pour servir vos ressources web plus rapidement aux utilisateurs situés à proximité.

Utilisez des requêtes HTTP asynchrones. Les requêtes HTTP asynchrones permettent à votre application de continuer à fonctionner pendant qu'une requête est en cours d'exécution.

**Questions sur la Résolution de Problèmes:**

* Questions

"Comment abordez-vous le débogage d'une application MERN ? Pouvez-vous donner un exemple spécifique d'un problème que vous avez résolu ?"

Je commence par l’identification du problème (tester l’ensemble du composants ou l’erreur a été introduise dans frontend)

Après maintenant je vais l’identifiée sur le code en utilisant les outils de débogage (Try/catch ou console.log)

Maintenant je le répare si je connais la solution, si c’est une nouvelle erreur pour moi, je le recherche sur l’internet ou la documentation concernée

"Quelle approche utilisez-vous pour optimiser les performances d'une application web ?"

* Optimiser le code s’il n’est pas optimiser(éviter les boucles inutiles et utiliser des fonctions de bibliothèque performantes)
* Utiliser des images optimisés
* Utilisez des CDN (pour servir vos ressources web plus rapidement aux utilisateurs situés à proximité.)
* Réponses

**Questions sur le Travail d'Équipe et la Communication:**

"Comment communiquez-vous avec les autres membres de votre équipe lors du développement d'un projet ?"

"Pouvez-vous donner un exemple de la façon dont vous avez contribué à la réussite d'un projet d'équipe ?"

**Questions sur l'Apprentissage Continu et la Curiosité Technique:**

"Comment vous tenez-vous à jour avec les nouvelles technologies et tendances en développement web ?"

"Quel a été le dernier sujet technique que vous avez appris par vous-même, et pourquoi l'avez-vous choisi ?"