TD1 – ALESSIO VINCI

Description des Programmes

1. prog1.js - Affichage d'un Message Simple

Le programme prog1.js est un script élémentaire en Node.js qui a pour but d'afficher un message simple sur la console. En utilisant la fonction console.log("hello world"), ce script constitue une introduction aux bases de la programmation en JavaScript et à l'utilisation de Node.js. En définissant le script package.json et lançant le script avec npm start, l'utilisateur peut observer la sortie directement dans la console, affirmant ainsi que Node.js fonctionne correctement.

---

2. prog2.js - Affichage des Informations du Processus

Le script prog2.js est conçu pour fournir des informations clés sur le processus Node.js en cours d'exécution. Il affiche des données essentielles telles que le PID (Process ID) du processus courant, le PID du processus parent, le chemin courant, le système d'exploitation, l'architecture du processeur, le chemin d'exécution de Node.js, et la version de Node.js. Grâce à l'objet global process, ce programme illustre comment accéder à ces informations. Cette fonctionnalité est particulièrement utile pour le débogage et l'analyse des performances, permettant aux développeurs de mieux comprendre l'environnement d'exécution de leur application.

---

3. prog3.js - Liste des Variables d'Environnement

Dans prog3.js, le script se concentre sur l'affichage des variables d'environnement disponibles pour le processus Node.js. En accédant à process.env, le programme extrait et affiche un ensemble de variables qui peuvent influencer le comportement de l'application. Les variables d'environnement sont souvent utilisées pour gérer des configurations spécifiques, telles que les clés d'API, les chemins d'accès aux fichiers, ou les paramètres de connexion à la base de données. Ce script met en évidence l'importance des variables d'environnement dans la configuration et la

personnalisation des applications, et il permet aux développeurs de visualiser rapidement les paramètres en cours d'utilisation.

---

4. prog4.js - Transmission de Variables d'Environnement

Le programme prog4.js illustre comment transmettre des variables d'environnement à un script Node.js. En utilisant la commande de ligne suivante, l'utilisateur peut définir les variables PORT et MODE : PORT=8080 MODE=debug node prog4.js. Ce script accède aux variables d'environnement définies et les affiche dans la console. Cela montre comment Node.js peut être intégré dans des environnements de déploiement flexibles où les configurations peuvent varier en fonction des besoins. Ce programme souligne l'importance de la communication entre le shell et le processus Node.js, permettant ainsi une personnalisation dynamique du comportement de l'application en fonction des variables d'environnement.

prog5.js - Attente et Affichage du Temps Écoulé

Le programme prog5.js est conçu pour illustrer l'utilisation de setTimeout et de process.hrtime.bigint(). L'objectif principal de ce script est de faire attendre le programme un certain nombre de millisecondes (défini par la variable x) avant d'afficher le temps écoulé. En premier lieu, le script enregistre le temps de départ en nanosecondes en utilisant process.hrtime.bigint(), puis utilise setTimeout pour exécuter une fonction de rappel après x millisecondes. À l'intérieur de cette fonction, le programme calcule le temps écoulé depuis le départ, le convertit en millisecondes, et l'affiche dans la console. Ce script démontre comment gérer le temps d'attente de manière asynchrone et comprend les bases de l'utilisation de callbacks en JavaScript.

prog6.js - Affichage Répété d'un Message de Bienvenue

Le programme prog6.js se concentre sur l'affichage d'un message de bienvenue toutes les 10 millisecondes. En utilisant setTimeout, le script définit une fonction afficherMessageBienvenue() qui, lorsqu'elle est appelée, affiche le message et se rappelle elle-même après 10 millisecondes. Cela crée une boucle d'affichage continu qui ne se termine que lorsque le programme est interrompu manuellement. Ce script illustre l'utilisation de la récursivité avec

des temporisateurs en JavaScript, permettant de réaliser des actions répétées de manière simple et efficace.

prog7.js - Gestion de l'Interruption avec Ctrl-C

Dans prog7.js, le programme traite l'interruption d'un processus par l'utilisateur via la combinaison de touches Ctrl-C. Lorsque cette interruption se produit, le programme doit gérer le signal SIGINT pour effectuer un nettoyage approprié ou afficher un message avant de quitter. En utilisant process.on('SIGINT', ...), le script peut écouter cet événement et exécuter une fonction qui affiche un message de fermeture ou effectue d'autres actions nécessaires avant de se terminer proprement. Ce script souligne l'importance de la gestion des événements dans les applications Node.js, en permettant une interruption contrôlée et une bonne expérience utilisateur.

```
setTimeout(()=>{
   console.log('Hey It is me !!');
}, 1000);
console.log('Why am I displayed first?');
```

In cette code le message 'Why am I displayed first?' affichè d'abord parce que le otre message doit attendre 1 sec.

prog8.js - Redirection du Flux d'Entrée vers le Flux de Sortie

Le programme prog8.js met en œuvre la redirection du flux d'entrée standard (stdin) vers le flux de sortie standard (stdout). En utilisant la méthode pipe, le script lit les données saisies par l'utilisateur dans la console et les réécrit immédiatement. Cela permet d'observer en temps réel ce qui est saisi, transformant ainsi le terminal en un écho de la saisie. Cette fonctionnalité est particulièrement utile pour le débogage et le développement de lignes de commande interactives, offrant une interface simple pour interagir avec l'utilisateur.

prog9.js - Affichage du Nombre de Caractères Saisis

Le programme prog9.js est conçu pour interagir avec l'utilisateur via la ligne de commande. Lorsqu'il est exécuté, il écoute les saisies de l'utilisateur et affiche en temps réel le nombre total de caractères qu'il a saisis. Cela se fait en utilisant

le flux d'entrée standard (stdin), qui permet de capturer les données saisies. À chaque fois que l'utilisateur entre du texte et appuie sur "Entrée", le programme compte le nombre de caractères de cette entrée et l'additionne à un compteur total. Ainsi, si un utilisateur saisit "hello", suivi de "bonjour", le programme affichera respectivement 6 et 8, ce qui reflète le nombre total de caractères saisis jusqu'à présent.

prog10.js - Utilisation de la Librairie Readline

Le script prog10.js introduit la librairie readline, qui améliore l'interaction avec l'utilisateur en fournissant un contrôle plus précis sur la saisie de texte. Ce programme fonctionne de manière similaire à prog9.js, mais avec des améliorations significatives. Il permet de lire les entrées ligne par ligne. Chaque fois qu'une ligne est saisie, le programme calcule le nombre de caractères de cette ligne et l'ajoute à un compteur total, affichant ainsi le nombre total de caractères saisis. Cette approche permet une meilleure gestion des entrées, car l'utilisateur peut voir le résultat après chaque ligne saisie, offrant ainsi une expérience plus interactive et conviviale.

prog11.js - Redirection de l'Entrée du Clavier dans un Fichier

Le programme prog11.js prend un tournant intéressant en redirigeant les entrées utilisateur vers un fichier. En utilisant la fonction createWriteStream, ce script crée un flux d'écriture qui permet de sauvegarder toutes les données saisies par l'utilisateur dans un fichier spécifié. Lorsque l'utilisateur tape quelque chose dans la ligne de commande, ces entrées sont capturées et écrites directement dans le fichier au lieu d'être affichées à l'écran. Cela peut être particulièrement utile pour les applications où il est nécessaire de conserver une trace des entrées utilisateur, comme pour les journaux ou les enregistrements d'activité.

prog12.js - Copie d'un Fichier

Le dernier programme, prog12.js, traite de la copie de fichiers en utilisant les flux. Ce script emploie la fonction createReadStream pour lire le contenu d'un fichier source, puis utilise la méthode pipe pour écrire ce contenu dans un fichier de destination. Grâce à cette méthode, le programme permet de copier efficacement le contenu d'un fichier à un autre sans avoir besoin de charger

l'intégralité du fichier en mémoire, ce qui est particulièrement bénéfique pour les fichiers volumineux. Cette fonctionnalité est essentielle dans de nombreux contextes où le transfert de données entre fichiers est nécessaire, comme dans les systèmes de gestion de fichiers ou les applications de sauvegarde.