Lab1 : Les modules de base de Node.js

Exercice 1 : Ecrire et lancer un premier programme node.js

1. Dans votre espace de travail, créer un fichier hello.js
2. Ouvrir le fichier hello.js et ajouter l'instruction suivante :

```
console.log("Hello World !");
```

3. Enregistrer le fichier hello.js.
4. Ouvrir le terminal dans le répertoire de travail puis taper la commande :

```
$ node hello.js
```

5. Remarquer l'affichage du message « Hello World ! » sur la console.
6. revenir au fichier hello.js et remplacer le code précédent par celui-ci pour tester quelques core modules de node.

```
var path = require('path');
var util = require('util');
var v8 = require('v8');

//Afficher le nom du fichier courant avec util.log et path.basename
util.log( path.basename(__filename) );

//Créer un chemin avec path.join puis affichage
var dirUploads = path.join(__dirname, 'www', 'files', 'uploads');
util.log(dirUploads);

//Afficher les statistiques de l'utilisation de la mémoire avec v8.
util.log(v8.getHeapStatistics());
```

7. Exécuter à nouveau votre programme et remarquer l'affichage de la date et l'heure devant les messages de log dans la console.

Exercice 2 : Interagir avec l'utilisateur à l'aide de stdin et stdout

1. Dans votre espace de travail, créer le fichier inout.js
2. Ouvrir le fichier créé et ajouter le code suivant :

```
//Initialisation d'un tableau avec quelques questions à poser
var questions = [
  "What is your name?",
  "What is your fav hobby?",
  "What is your preferred programming language?"
];

//Préparation d'un tableau pour stocker les réponses de l'utilisateur
var answers = [];

// Préparation de la fonction ask qui permet d'afficher une question à l'écran
function ask(i) {
  process.stdout.write(`\n\n ${questions[i]}`);
  process.stdout.write("  >  ");
}

//Préparation d'un écouteur pour intercepter la saisie des réponses
process.stdin.on('data', function(data) {

        answers.push(data.toString().trim());
```

```
        if (answers.length < questions.length) {
                ask(answers.length);
        } else {
                //Annoncer la fin du programme
                process.exit();
        }

});

//Ecouter la fin du programme pour afficher les résultats
process.on('exit', function() {

        process.stdout.write("\n\n\n\n");

        process.stdout.write(`Go ${answers[1]} ${answers[0]} you can finish writing
${answers[2]} later`);

        process.stdout.write("\n\n\n\n");

});

//Appel de la fonction ask pour poser la première question
ask(0);
```

3. Lancer le programme et répondre aux question pour afficher le résultat résultat final.

Exercice 3 : Lancer et intercepter des évènements personnalisés avec EventEmitter

1. Dans votre espace de travail, créer un fichier event.js
2. Copier le code suivant dans votre fichier.

```
//importer les core modules eventEmitter et util
var EventEmitter = require('events').EventEmitter;
var util = require('util');

//créer un constructeur d'objets Person
var Person = function(name) {
        this.name = name;
};

//Hériter les propriétés et méthodes de EventEmitter dans Person
util.inherits(Person, EventEmitter);

//Créer une instance d'objet Person
var ben = new Person("Ben Franklin");

//Préparer un écouteur et une callback function pour l'événement speak avec on.
ben.on('speak', function(said) {

        console.log(`${this.name}: ${said}`);

});

//Emettre l'événement speak avec emit.
ben.emit('speak', "You may delay, but time will not.");
```

3. Lancer le programme et remarquer l'affichage du message `"You may delay, but time will not."`

Exercice 4 : Lister le contenu d'un répertoire

1. Dans votre espace de travail, créer un répertoire /fs
2. Copier le répertoire /lib joint à ce lab sous le répertoire /fs
3. Créer un fichier list.js sous /fs.
4. Copier le code suivant dans le fichier list.js

```
var fs = require("fs");
fs.readdir('./lib', function(err, files) {
        if (err) {
                throw err;
        }
        console.log(files);
});
console.log("Reading Files...");
```

5. Lancer votre programme et vérifier l'affichage de la liste des fichiers.

Exercice 5 : Lire le contenu des fichiers

1. Dans le répertoire /fs créé dans la question précédente, vérifier l'existance du sous-répertoire /li puis ajouter un fichier nommé read.js
2. Copier le code suivant dans le nouveau fichier créé

```
var fs = require("fs");
var path = require("path");

fs.readdir("./lib", function(err, files) {

        files.forEach(function(fileName) {
                var file = path.join(__dirname, "lib", fileName);
                var stats = fs.statSync(file);
                if(stats.isFile() && fileName !== ".DS_Store") {

                        fs.readFile(file, "UTF-8", function(err, contents) {

                                console.log(contents);

                        });

                }
        });

});
```

3. Lancer le programme et vérifier l'affichage du contenu des fichiers.

Exercice 6 : Créer et Ecrire dans un fichier

1. Dans l répertoire /fs, ajouter un fichier nommé create.js
2. Copier le code suivant dans le nouveau fichier créé

```
var fs = require("fs");

var md = `

Sample Markdown Title
=====================

Sample subtitle
---------------

* point
* point
* point

`;

fs.writeFile("sample.md", md.trim(), function(err) {

        console.log("File Created");
});
```

3. Lancer le programme et vérifier la création du fichier avec le texte affecté à la variable md.

Exercice 7 : Créer un serveur web simple

1. Dans votre espace de travail, créer un fichier server.js
2. Importer le module http à l'aide de require.

```
var http = require("http");
```

3. Créer un serveur

Nous utilisons l'instance http créée et appelons la méthode **http.createServer()** pour créer une instance de serveur, puis nous la lions au port 8081 à l'aide de la méthode **listen** associée à l'instance du serveur.

Passez une fonction avec les paramètres de requête et de réponse. Ecrivez l'exemple d'implémentation pour toujours retourner "Hello World".

```
http.createServer(function (request, response) {
        response.writeHead(200, {'Content-Type': 'text/plain'});
        response.end('Hello World\n');
}).listen(8081);

// Affichage d'un message sur la console pour indiquer le lancement du serveur
console.log('Server running at http://127.0.0.1:8081/');
```

4. Lancer le serveur :

A partir de votre terminal (invite de commande), lancer la commande :

```
$ node server.js
```

Ceci permettra de lancer le serveur créé dans server.js

Dans le terminal, vous devez voir affiché le message suivant :

```
Server running at http://127.0.0.1:8081/
```

5. Accéder à l'aide de votre navigateur à l'adresse http://127.0.0.1:8081/ pour voir le résultat :