Etablissement : ISET Charguia	**Département :** Technologies de l'Informatique
Matière : Atelier Programmation côté serveur	**Classe :** DSI3
Année Universitaire : 2020 - 2021	

TP n° 2 : Node.JS – Module, package, NPM

Objectifs du TP :
- Maitrisez les modules
- Utiliser les packages
- Exploitez NPM

N.B : L'ensemble des exercices doivent être placés dans un répertoire nommé « TP2 » placé sous un répertoire portant votre nom, prénom et classe.

Partie 1

1. **Exercice 1**

Présentation

- Le noyau de Node.js est tout petit.
- Il existe des milliers de modules qui offrent des fonctionnalités variées :
  - gestion des fichiers uploadés
  - connexion aux bases de données MySQL, Redis, MongoDB
  - utilisation de frameworks
  - utilisation de systèmes de templates
  - gestion de la communication temps réel avec le visiteur
  - etc.
- Il y a à peu près tout ce dont on peut rêver et de nouveaux modules apparaissent chaque jour.

Module intégré

- Un module c'est une bibliothèque de fonctions (ou méthodes) et possiblement d'autres classes incluses.
- Un module est une API : Application Programming Interface.
- Node.js dispose de modules dit « intégrés » qui sont présents dès l'installation de Node.js,comme « http », « url », etc.

https://nodejs.org/api/

Inclure un module

- Inclure un module c'est créer un objet qui permet l'accès aux méthodes et aux classes inclusesdu module.
- Pour inclure un module intégré, on utilise la fonction require() avec le nom du module.

➢ **Exemple : le module http**

- https://nodejs.org/api/http.html
- On voit dans la documentation toutes les caractéristiques du modules : les fonctions et classesobjets auxquels il donne accès.
- Par exemple : la clase http.Agent, la fonction createServer().

➢ **utilisation**

```
const http = require('http')
```

- L'objet « http » permet d'accéder aux méthodes et aux classes prédéfinies.
- On peut écrire un accès à la méthode createServer :

```
http.createServer(…)
```

- On peut instancier un objet de la classe http.Agent :

```
const agent = new http.Agent({ keepAlive: true });
```

- le feu reste 3 secondes au rouge puis on relance un cycle Indice

Les Callbacks

Exercice 1 : Réalisez un script contenant une fonction **output** qui accepte une fonction callback **result** qui permet de faire l'addition de deux entier. La fonction **output** affiche : « **le résultat est …. »**

Exercice 2 : Réalisez maintenant une fonction **compute (x1,x2, mycallback)**, qui accepte une fonction callback qui réalise le produit de deux entiers.

Exercice 3 : soit le code source ci-dessous, exécutez-le et dites s'il sagit d'un code bloquant ou non bloquant :

```
var fs = require("fs");
var data = fs.readFileSync('input.txt');

console.log(data.toString());
console.log("Program Ended");
```

Transformez ce code en utilisant une fonction de type callback pourque ça devient non bloquant. Vérifiez-le.

Javascript Promises

Rappel : la forme d'une promesse en JavaScript est la suivante :

```
a(() => {
  b(() => {
    c(() => {
      d(() => {
        // and so on ...
      });
    });
  });
});
```

qui devient comme suit:

```
Promise.resolve()
  .then(a)
  .then(b)
  .then(c)
  .then(d)
  .catch(console.error);
```

Syntaxe

```
1. asyncFunc()
2. .then(value => { /* success */ })
3. .catch(error => { /* failure */ });
```

Partie 2 : Modules externes

Exercice 1 (Module utilisateur)

Créez un module Node.js appelé "mathUtils" qui expose deux fonctions :

1. Une fonction nommée "add" qui prend deux nombres en paramètre et retourne leur somme.
2. Une fonction nommée "multiply" qui prend deux nombres en paramètre et retourne leur produit.

Ensuite, créez un fichier JavaScript appelé "main.js" qui utilise le module "mathUtils" pour effectuer les opérations suivantes :

1. Importez le module "mathUtils" dans votre fichier "main.js".
2. Utilisez la fonction "add" pour additionner les nombres 5 et 3, puis affichez le résultat.
3. Utilisez la fonction "multiply" pour multiplier les nombres 4 et 2, puis affichez le résultat.

Instructions :

1. Créez un nouveau répertoire pour cet exercice.
2. Créez un fichier "mathUtils.js" dans ce répertoire et définissez les deux fonctions "add" et "multiply" à l'intérieur.
3. Créez un fichier "main.js" dans le même répertoire et importez le module "mathUtils" à l'aide de la fonction require.
4. Utilisez les fonctions "add" et "multiply" pour effectuer les calculs demandés.
5. Exécutez le fichier "main.js" à l'aide de Node.js et vérifiez si les résultats sont correctement affichés.

Voici le code de départ pour vous aider à démarrer :

```
// mathUtils.js
function add(a, b) {
// TODO: Implémentez la fonction d'addition
}
function multiply(a, b) {
// TODO: Implémentez la fonction de multiplication
}
```

Exercice 2 : (Module externe moment)

*Le module **Moment.js** est une bibliothèque populaire en Node.js qui permet de manipuler, formater et analyser des dates et des heures.*

Installer et utiliser le module moment pour :

1. Afficher la date et l'heure actuelle.
2. Afficher combien de temps nous sépare d'une date ultérieur telle que : 27/02/2015
3. Afficher combien de minutes sont passées du début du jour actuel
4. Afficher combien de minutes restent-il avant la fin de l'heure actuelle.

Exercice 3 (Module externe Sharp)

Sharp *: Une bibliothèque pour la manipulation d'images en Node.js, offrant des fonctionnalités de redimensionnement, de recadrage, de compression, etc.*

Etant donnée une image ‘input.jpg’ utiliser sharp (la méthode resize(…)) afin de redimensionner cette dernière et fournir une image résultat ‘output.jpg’. L’image aura une largeur de 800 pixels et une hauteur proportionnelle.

- Le message : " l’image a été redimensionnée avec succès ! » est affiché en cas de succès et des infos seront affichées.
- Sinon un message d’erreur sera affiché.