==========Partie 3==========

Module URL Node.js

1) Le module d'URL intégré

Le module URL divise une adresse Web en parties lisibles.

Pour inclure le module URL, utilisez la **`require()`** méthode :

```
var url = require('url');
```

Analysez une adresse avec la `url.parse()` méthode et elle renverra un objet URL avec chaque partie de l'adresse en tant que propriétés.

Le code **`url.parse(adr, true)`** est utilisé pour analyser une URL et en extraire les différentes parties qui la composent, telles que le protocole, l'hôte, le chemin, les paramètres de requête, etc.

Voici une explication détaillée de chaque paramètre de la méthode **`url.parse()`** :

- **`adr`** : Il s'agit de l'URL que nous souhaitons analyser.
- **`true`** : Ce deuxième paramètre est un booléen qui indique si les paramètres de la requête doivent être analysés en tant qu'objet (**`true`**) ou en tant que chaîne de requête (**`false`**).

La méthode **`url.parse()`** renvoie un objet qui contient les différentes parties de l'URL analysée. Voici les principales propriétés de l'objet renvoyé :

- **`protocol`** : Le protocole utilisé (http, https, ftp, etc.)
- **`host`** : L'hôte (nom de domaine ou adresse IP) de l'URL.
- **`pathname`** : Le chemin de l'URL.
- **`query`** : Les paramètres de la requête, soit sous forme d'une chaîne de requête (si **`true`** est **`false`**), soit sous forme d'un objet (si **`true`** est **`true`**).
- **`hash`** : le fragment d'URL, également appelé ancre ou hash, fait référence à la partie optionnelle d'une URL qui suit le symbole "#" et qui peut être utilisée pour identifier une section spécifique d'une ressource Web.

En résumé, le code **`url.parse(adr, true)`** analyse l'URL **`adr`** en tant qu'objet et renvoie un objet contenant les différentes parties de l'URL.

Exemple

Divisez une adresse Web en parties lisibles :

```
var url = require('url');
var adr = 'http://localhost:8084/default.htm?year=2017&month=february#section2';
var q = url.parse(adr, true);

console.log(q.protocol); //returns 'http:'
console.log(q.host); //returns 'localhost:8084'
console.log(q.pathname); //returns '/default.htm'
console.log(q.search); //returns '?year=2017&month=february'

var qdata = q.query; //returns an object: { year: 2017, month: 'february' }
console.log(qdata.month); //returns 'february'
console.log(qdata.year); //returns '2017'

console.log(q.hash); //returns '#section2:'
```

2) Serveur de fichiers Node.js

Nous savons maintenant comment analyser la chaîne de requête et, dans le chapitre précédent, nous avons appris à faire en sorte que Node.js se comporte comme un serveur de fichiers. Combinons les deux, et servons le dossier demandé par le client.

Créez deux fichiers html et enregistrez-les dans le même dossier que vos fichiers node.js.

été.html

```
<!DOCTYPE html>
<html>
<body>
<h1>Summer</h1>
<p>I love the sun!</p>
</body>
</html>
```

hiver.html

```
<!DOCTYPE html>
<html>
<body>
<h1>Winter</h1>
<p>I love the snow!</p>
</body>
</html>
```

Créez un fichier **t3.js** Node.js qui ouvre le fichier demandé et renvoie le contenu au client. Si quelque chose ne va pas, lancez une erreur 404 :

t3.js :

```
var http = require('http');
var url = require('url');
var fs = require('fs');

http.createServer(function (req, res) {
  var q = url.parse(req.url, true);
  var filename = "." + q.pathname;
  //Le point (".") représente le répertoire courant,
  // donc en ajoutant le point devant le chemin relatif (pathname)
  //extrait de l'URL, cela crée un chemin de fichier relatif
  //qui commence à partir du répertoire courant.
```

```
  fs.readFile(filename, function(err, data) {
    if (err) {
      res.writeHead(404, {'Content-Type': 'text/html'});
      return res.end("404 Not Found");
    }
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });

}).listen(8084);
console.log("Le serveur est en cours d'execution");
console.log("l'adresse : http://localhost:8084/summer.html ");
console.log("l'adresse : http://localhost:8084/winter.html ");
```

Pensez à initier le fichier :

Lancez t2.js :

```
C:\Users\Your Name>node t2.js
```

Si vous avez suivi les mêmes étapes sur votre ordinateur, vous devriez voir deux résultats différents lors de l'ouverture de ces deux adresses :

http://localhost:8084/summer.html

Produira ce résultat:

Summer

I love the sun!

http://localhost:8084/winter.html

Produira ce résultat:

Winter

I love the snow!

SERIE EXERCICES

I) SUR le système de fichiers Node.js :

1) Créez un fichier texte "message.txt" et écrivez-y "Bonjour Node.js !" en utilisant la méthode fs.writeFile().
2) Lisez le contenu du fichier "message.txt" à l'aide de la méthode fs.readFile() et affichez-le dans la console.
3) Créez un dossier "documents" avec la méthode fs.mkdir().
4) Copiez le fichier "message.txt" dans le dossier "documents" avec la méthode fs.copyFile().
5) Supprimez le fichier "message.txt" avec la méthode fs.unlink().
6) Renommez le fichier "documents/message.txt" en "documents/nouveau-message.txt" avec la méthode fs.rename().
7) Affichez la liste des fichiers et dossiers présents dans le dossier "documents" avec la méthode fs.readdir().
8) Créez un fichier "data.json" contenant un objet avec des propriétés et des valeurs de votre choix avec la méthode fs.writeFile().
9) Lisez le contenu du fichier "data.json" à l'aide de la méthode fs.readFile() et affichez-le dans la console.
10) Modifiez une propriété de l'objet contenu dans le fichier "data.json" avec la méthode fs.readFile(), puis écrivez les modifications dans le fichier avec la méthode fs.writeFile().

II) SUR 'URL Node.js :

1) Utiliser le module 'url' pour extraire les paramètres de requête d'une URL donnée .
2) Utiliser le module 'url' pour extraire le chemin d'une URL donnée.
3) Utiliser le module 'url' pour résoudre une URL relative en une URL absolue.
4) Utiliser le module 'url' pour parser une URL donnée en ses différentes parties (protocole, hôte, chemin, etc.).
5) Utiliser le module 'url' pour générer une URL à partir de ses différentes parties (protocole, hôte, chemin, etc.).
6) Utiliser le module 'querystring' pour créer une chaîne de requête à partir d'un objet JavaScript.
7) Utiliser le module 'querystring' pour parser une chaîne de requête en un objet JavaScript.
8) Utiliser le module 'url' pour ajouter des paramètres de requête à une URL existante.
9) Utiliser le module 'url' pour extraire le nom de domaine d'une URL donnée.
10) Utiliser le module 'url' pour vérifier si une URL est absolue ou relative.

==========Partie 4==========

NPM Node.js

1) Qu'est-ce que le NPM ?

NPM est un gestionnaire de packages pour les packages Node.js ou les modules si vous le souhaitez.

www.npmjs.com héberge des milliers de packages gratuits à télécharger et à utiliser.

Le programme NPM est installé sur votre ordinateur lorsque vous installez Node.js

NPM est déjà prêt à fonctionner sur votre ordinateur !

2) Qu'est-ce qu'un forfait ?

Un package dans Node.js contient tous les fichiers dont vous avez besoin pour un module.

Les modules sont des bibliothèques JavaScript que vous pouvez inclure dans votre projet.

3) Télécharger un package

Le téléchargement d'un package est très simple.
Ouvrez l'interface de ligne de commande et dites à NPM de télécharger le package souhaité.
Je souhaite télécharger un package appelé "majuscule":

Télécharger "majuscule":

```
C:\Users\Your Name>npm install upper-case
```

Vous avez maintenant téléchargé et installé votre premier package !

NPM crée un dossier nommé **"node_modules"**, où le package sera placé. Tous les packages que vous installerez à l'avenir seront placés dans ce dossier.

Mon projet a maintenant une structure de dossiers comme celle-ci :

```
C:\Users\My Name\node_modules\upper-case
```

3) Utilisation d'un package

Une fois le package installé, il est prêt à être utilisé.

Incluez le package "majuscules" de la même manière que vous incluez n'importe quel autre module :

```
var uc = require('upper-case');
```

Créez un fichier Node.js qui convertira la sortie "Hello World!" en majuscules :

Exemple

```
var http = require('http');
var uc = require('upper-case');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write(uc.upperCase("hello horld! Node js et javascript"));
  res.end();
}).listen(8084);
```

Enregistrez le code ci-dessus dans un fichier appelé "t1.js", et lancez le fichier :

Lancez t1:

```
C:\Users\Your Name>node t1
```

Si vous avez suivi les mêmes étapes sur votre ordinateur, vous verrez le même résultat que l'exemple : http://localhost:8084