TD7 : node.js comme client, fs, request, jsdom, url-parse, programmation fonctionnelle.

Node.js est (beaucoup) plus qu'un simple outil pour réaliser un serveur. Dans ce TD, vous allez utiliser node.js comme un client web et réaliser des opérations d'analyse de pages Web et de téléchargement. Si vous utilisez node >= 4, utilisez la syntaxe des fonctions avec ***() => { }*** plutôt que ***function***… Vous ne proposerez que des solutions de programmation fonctionnelles pour ce TD.

Packages : **fs**, **request**, **jsdom**, **url-parse**

1. Etudiez le package 'fs' qui permet de réaliser des entrées/sorties fichier javascript. A moins de cas de force majeure (à justifier), utilisez toujours les méthodes asynchrones. En particulier, regardez les méthodes qui concernent les ***streams*** (***createWriteStream***, ***pipe*** etc.).

2. Chargez et étudiez le package 'request'. Ce Package permet de réaliser des requêtes HTTP très simplement : d'une certaine manière, vous pouvez réaliser les requêtes Ajax d'une manière plus simple et SANS les limitations d'accès d'Ajax (puisque node.js ne fonctionne pas dans un navigateur). Vous pouvez donc faire des requêtes sur n'importe quel serveur. Exemple :

```
var request = require('request');
request('http://uwamiz.com/rongeurs/hamster-2',  (error, response, body) => {
  if (!error && response.statusCode == 200) {
    console.log(body) // Affiche le HTML des rongeurs.
  }
})
```

3. En utilisant 'fs' et 'request', réalisez la fonction ***chargement(url, nomFichier, errCb)*** qui réalise le chargement de l'URL et sauvegarde le fichier obtenu sous le nom ***nomFichier***. ***errCb*** est une fonction de la forme ***function errCb(err, nomFichier)*** de gestion d'erreur. Evidemment, cette fonction ***chargement*** est asynchrone. Attention, cette fonction doit être capable de télécharger une image (par exemple). Consultez le protocole HTTP pour voir comment les images sont transférées. Indice : utilisez les ***streams*** (pipe) et les événements (**on(…)**).

4. On souhaite lire une page web, détecter les images de cette page et sauvegarder les images localement. Node.js interprète du javascript en dehors d'un navigateur, il n'est donc pas possible d'avoir simplement accès à un DOM. Vous allez utiliser une librairie d'analyse de page Web qui recrée un DOM en mémoire. Une fois ce DOM créé, vous pourrez détecter les images en cherchant les balise <IMG> (et donc les SRC) et charger les images localement. Commencez par récupérer 'jsdom' par ***npm*** et étudiez les exemples. Réalisez un programme node.js qui lit une page Web et affiche les noms des images de la page dans la console.

5. Comme vous l'avez sans doute remarqué, les noms des images sont souvent des noms locaux : ils doivent être préfixés par le nom de domaine et sans doute le chemin relatif de la page pour pouvoir être téléchargés par votre fonction chargement. Il est donc nécessaire d'analyser les chemins de l'URL de la page et des images. Utilisez le package ***url-parse*** pour faire ce travail.

6. Enfin, réalisez une fonction ***recupereImages(url, prefixe, errCb)*** qui charge la page url, analyse la page et télécharge localement les images de la page. Les noms locaux seront préfixés par le paramètre donné. Par exemple si préfixe est égale à 'im', alors les images auront les noms 'im0.jpg', 'im1.gif ', etc..

7. Testez votre méthode sur une page qui contient beaucoup d'images (évitez ***google.com*** (!) et ne prenez pas tous la même page).