302 – Scripting et CLI

node.js

WIK-NJS302

Durée estimée : 2h (hors TP final)

Intervenant : Jeremy Trufier <jeremy@wikodit.fr>

ynov

wikodit

WIK-NJS

Programme nodeJS

301 – Introduction
302 – Scripting et CLI
303 – Express.js
304 – MVC Frameworks
(305 – Tests unitaires)

1XX – 1er année (pas de notion d'algorithmie)
2XX – 2e année (notions d'algorithmie succintes)
3XX – 3e année (rappels et pratique, niveau moyen d'algorithmie)
4XX – 4e année (concepts avancés, niveau avancé d'algorithmie)
5XX – 5e année (approfondissement experts)

Au préalable

Scripts habituels

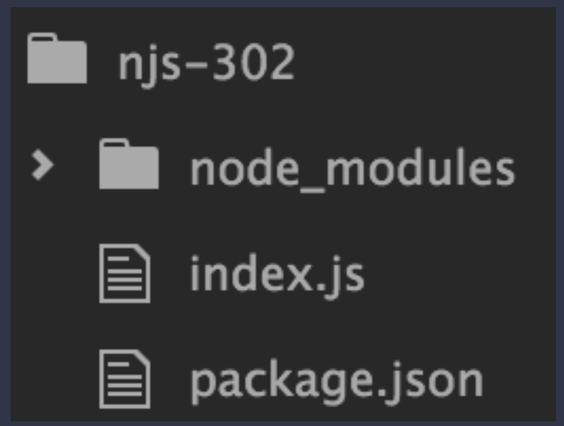
- Bash
- Fastidieux
- Gestion I/O catastrophique
- Procédural
- Gestion d'évènements compliqués
- Pas de library externe

nodeJS en scripting

- Plus haut niveau
- Asynchrone (parallélisation des traitements)
- Très bonne gestion d'évènement
- Accès à un grand nombre de modules
  - FS
  - S3
  - Databases
  - ...

Une commande en node

1 - Arborescence



2 - index.js

```
#!/usr/bin/env node
console.log('Hello world!')
```

3 - package.json

```
{
  "name": "njs-302",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "bin": {
    "mycommand": "./index.js"
  }
}
```

4 - Terminal

```
$ cd njs-302
$ npm install -g
$ mycommand
Hello world!
```

Les modules utiles

Module : commander

Gérer les options d'une commande facilement

```
#!/usr/bin/env node
const program = require('commander')

// Configuration des paramètres attendus
program
  .version('1.0.0')
  .option('-w, --world', 'Show hello world')
  .option('-a, --all', 'Show hello all')
  .option('-s, --someone [name]', 'Say hi to someone')

// On parse (convertit en format utilisable) les options
// fonction synchrone
program.parse(process.argv)

// Maintenant on peut les utiliser
if (program.world) {
  console.log('Hello world!')
} else if (program.all) {
  console.log('Hello all!')
} else if (program.someone) {
  console.log(`Hello ${program.someone}!`)
} else {
  program.help()
}
```

Doc : https://github.com/tj/commander.js/

Module : inquirer

Demander des choses à
l'utilisateur de façon stylée !

```
const inquirer = require('inquirer')

inquirer.prompt([
  {
    type: 'input',
    message: 'Entrez votre nom d\'utilisateur',
    name: 'username'
  }, {
    type: 'password',
    message: 'Entrez votre mot de passe',
    name: 'password'
  }, {
    type: 'checkbox',
    message: 'Que voulez-vous sauvegarder ?',
    name: 'foldersToSave',
    choices: [
      'Mes documents',
      'Mon bureau',
      'Ma musique'
    ]
  }
]).then((answers) => {
  console.log(answers)
})
```

Doc : https://github.com/SBoudrias/Inquirer.js

Manipulation du FileSystem

```
const fs = require('fs')

try {
  // Écrire un fichier
  fs.writeFile('message.txt', 'Bonjour !', (err) => {
    if (err) throw err
    console.log('Fichier écrit')
  })

  // Lire un fichier
  fs.readFile('message.txt', 'utf8', (err, data) => {
    if (err) throw err
    console.log('Données du fichier : ' + data)
  })
} catch (err) {
  console.error('ERR > ', err)
}
```

Et aussi :

- fs.rename(oldPath, newPath, callback)
- fs.rmdir(path, callback)
- fs.mkdir(path, callback)
- ...

Doc : https://nodejs.org/api/fs.html

Un module qui encapsule les méthodes dans des promesses existe et s'appelle "fs-promise"

Execution de commandes CLI

Commandes simples

```
const exec = require('child_process').exec;

exec('ls -lh ./', function(err, stdout, stderr) {
  if(err) {
    return console.error('ERR > ', err)
  }
  console.log(`stdout: ${stdout}`)
  console.log(`stderr: ${stderr}`)
});
```

Commandes qui renvoient beaucoup de données

```
const spawn = require('child_process').spawn

let ls = spawn('ls', [ '-lh', './' ])

ls.stdout.on('data', (data) => {
  console.log(`stdout: ${data}`)
})

ls.stderr.on('data', (data) => {
  console.log(`stderr: ${data}`)
})

ls.on( 'close', (code) => {
  console.log(`process terminé code ${code}`)
})
```

Autres modules utiles

- lodash => Bibliothèque de fonction utiles (équivalent de underscore.js)
- superagent => Permet de faire des requêtes HTTP
- chalk => Permet de faire de la coloration syntaxique dans la console (pour les artistes)
- progress => Une barre de progression en ASCII art
- blessed-contrib => Un dashboard entier en ASCII art
- co-prompt => Prompt bloquant
- moment => Pour la gestion des dates de façon extrêmement puissante et simplifiée
- nodemailer => Envoi d'email
- cli-table => Des tableaux dans le terminal

Utile : https://github.com/sindresorhus/awesome-nodejs

TP CLI

TP : Objectifs

Les notions suivantes doivent être acquises et utilisées

- Arguments de la commande (commander)
- Programme interactif (inquirer)
- Utilisation des promesses/asynchrone/évènements
- Manipulation de fichiers
- Manipulation de base Sqlite

TP : Sujets libre

- ask-me-something : prend un thème en option (avec --theme histoire), pose des questions de culture G
- tell-me : Synthèse vocale, prend une voie en option, et un texte ou mode interactif avec -i
- twit-img : Récupère les images récentes postées sur twitter, selon un hashtag passé en paramètre et d'autres options
- html-extract : Extraire des données de fichiers HTML (35000 fichiers à traiter)
- marvel-hero : Se connecte à l'API de marvel, pour extraire des données selon les paramètres demandés
- manga-crawl : Download les images d'un manga en ligne, track des mangas, affiche les nouveautés
- sqlite-orm : Permet d'utiliser SQLite en ligne de commande avec une interface simple

TP individuel

Félicitations !!

Cours WIK-NJS-302 burned :)