Programmation avancée

Node.js - MongoDB

Sylvie Trouilhet - Jean-Marie Pecatte

S4 – UE2

Plan

1. Faire un projet node.js
2. Tests unitaires avec Mocha

Première partie :
faire un projet

Projet node : initialisation

- Créer un répertoire **`testProjet`** et se placer dans ce répertoire
- Créer un fichier vide **`index.js`**
- Lancer la commande de création du module

```
npm init
```

Et répondre aux questions (nom du projet : tp3, version : 0.1.0, point de départ : index.js, etc)

→ Le fichier **`package.json`** est automatiquement créé

```
{
  "name": "tp3",
  "version": "0.1.0",
  "description": "projet node",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "st",
  "license": "LGPL"
}
```

Projet node : ajout de modules

- Installez **en local** les modules que votre projet utilise :

```
npm install --save express
```

→ L'option **`--save`** sert à gérer les dépendances

→ le module est dans le répertoire **`node_modules`**

NB : option –g pour une installation en **`global`**

- Les dépendances sont les fichiers qui sont liés à votre projet

- Elles sont indiquées dans package.json

```
"dependencies": {    "express": "^4.16.2"
                }
```

→ accepte les changements mineurs de version

Automatisation des tests : framework Mocha

- Framework Mocha : https://www.npmjs.com/package/mocha
- Installation (locale au projet) :

```
npm install --save-dev mocha
```

- Enregistrer les fichiers de test dans le répertoire **`test`**
- Commande **`node node_modules\mocha\bin\mocha`**

→ Ajout dans **`package.json`** de

```
"devDependencies": {
                "mocha": "^5.0.0"  }
```

- Créer un fichier vide **`testUtilitaire.js`**

Pour lancer le projet

- L'attribut **"scripts"**
  (https://docs.npmjs.com/misc/scripts)

- Completer **scripts** dans **package.json :**

```
"scripts": {
  "test": "mocha"  ,
  "start": "node index.js"
}
```

NB : npm test→ comme node node_modules\mocha\bin\mocha

- Compléter le fichier index.js :
  `console.log("bonjour");`

- Exécuter **`npm start`** → comme **`node index.js`**

- Créer un répertoire **lib**
- Ajouter le fichier **utilitaire.js**

```
function quoi(v1)
    {
    let n=1,f=1;
    while (f < v1) f*=++n;
    return f==v1;
    }
    module.exports.quoi = quoi;
```

Lors de la mise au point

- Le module **nodemon** : relance l'exécution du fichier à chaque modification
- Installer le module : **`npm install nodemon`**

- Remplacer **node** par **nodemon** dans **`package.json`**

Deuxième partie

Tests unitaires avec Mocha

Bug informatique…

- Le vol 501, vol inaugural de la fusée Ariane 5, a eu lieu le 4 juin 1996 et s'est soldé par un échec.
- La fusée s'est brisée et a explosé en vol, 40 secondes après le décollage, à la suite d'une panne du système de navigation.

- Coût : 370 millions de dollars, ce qui en fait le bug informatique le plus coûteux de l'histoire.

Les tests

- Tests unitaires : tester **chaque fonction** individuellement.

→ rédiger un ou plusieurs tests unitaires qui permettent de valider le comportement de la fonction, même en lui passant des paramètres incorrects (null, …)

- Test d'intégration : vérifier que tous les **éléments marchent ensemble**, et que ce qu'ils font est bien ce qui a été défini.

Tests unitaires

- Fonctionnement classique en 4 phases
- **Initialisation** (set up) : définition d'un environnement de test complètement reproductible
- **Exercice** : le module à tester est exécuté
- **Vérification** (assert) : comparaison des résultats obtenus avec un vecteur de résultats défini. Ces tests définissent le résultat du test : **`SUCCESS`** ou **`FAILURE`**
- **Désactivation** (clean up) : désinstallation pour retrouver l'état initial du système.
- Tous les tests doivent être indépendants et reproductibles unitairement.

Automatisation des tests : framework Mocha

```
describe
('Test du fichier XX', function()
      {
      // le fichier xx contient la fonction quoi
      describe
      ('test de quoi', function()
            {
            // vérification avec le module assert
            }
      );
      }
);
```

Automatisation des tests : framework Mocha

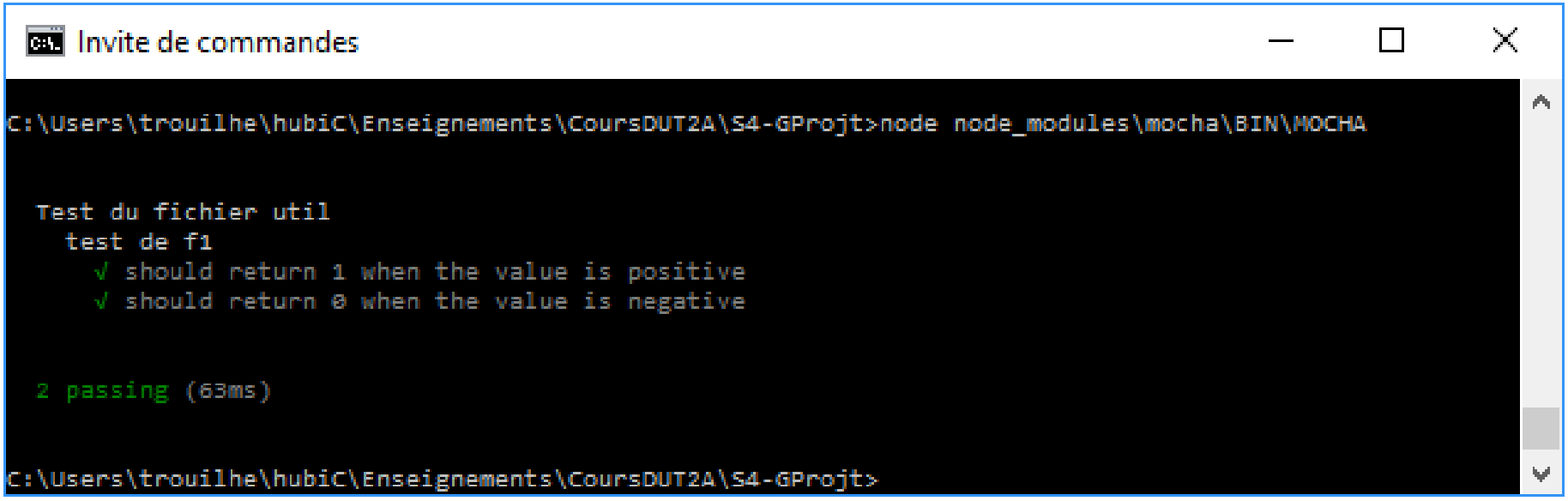
- **`ok(value, [msg])`** : l'expression est vraie
- **`equal(actual, expected, [msg])`** : deux expressions sont égales
- **`notEqual(actual, expected, [msg])`**

Automatisation des tests : framework Mocha

```
const assert = require('assert');
const m=require('../lib/utilitaire.js');

describe
('Test du fichier util', function()
       { describe
         ('test de f1', function()
           {
           it('should return true when …',
          function() {assert.equal(true, m.quoi(120));});
           it('should return false when …',
          function() {assert.equal(false, m.quoi(100));});
           }
        );
      }
);
```

Automatisation des tests : framework Mocha



Automatisation des tests : framework Mocha

- initialisation :

  `before`, qui sera exécuté avant la série de tests

  `beforeEach`, qui sera exécuté avant chaque test

- désactivation :

  `afterEach`, qui sera exécuté après chaque test

  `after`, qui sera exécuté après la série de tests

- Faire des tests complets pour la fonction du jeu feuille-pierre-ciseau
- Finaliser l'application pour que les joueurs puissent rejouer