



DOCUMENT ELEVE : JAVASCRIPT

Introduction au test

Objectif

A l'issue de cette activité, vous serez capable de :

- Ré-utiliser les notions algorithmique vues dans les premières semaines (conditions, boucles, tableaux, objets, fonctions, ...),
- Installer *npm* et les dépendances d'un projet
- Utiliser *npm* pour lancer une tâche
- Comprendre l'approche Test Driven Development (TDD)

Vous allez, dans cette progression, acquérir la compétence « **introduction au développement guidé par les tests** ». L'acquisition de ces compétences s'effectuera par la réalisation en Javascript de deux mini projets, prétextes à l'utilisation de tests.

Pré-requis

Bases du langage Javascript, algorithmique de base.

Installation des outils et préparation

Récupérez sur le drive S: le code du projet js-test.

On se garde en général dans un projet que le code spécifique à ce projet. L'ensemble des "briques" (code externe, librairies, etc) nécessaires à un projet est ce que l'on appelle les dépendances du projet.

Il existe plusieurs outils pour gérer les dépendances et l'"écosystème" de ces outils est fortement lié au langage de programmation utilisé. Pour Javascript, npm (Node.js Package Manager) est un standard de fait.

Installez *npm* en suivant les instructions depuis le site, <https://www.npmjs.com/> (vous pouvez également trouver sur le Web des introductions à npm - et Node.js¹ - mais ce n'est pas nécessaire pour ce projet qui ne vise qu'à une première introduction).

Une fois *npm* installé, ouvrez une ligne de commande dans le répertoire du projet (qui contient le fichier *package.json*) et exécutez "*npm install*". Cela va installer les dépendances du projet, qui sont décrites dans le fichier *package.json* (qui décrit également d'autres aspects du projet : ouvrez ce fichier pour voir qu'il contient un objet dans la notation Javascript).

Vous remarquerez la création du répertoire *node_modules*, qui contient toutes les dépendances (il y en a plus que celles indiquées dans le projet car chaque dépendance peut avoir à son tour des dépendances...)

Tester, corriger, recommencer

Le principe du test est bien évidemment de s'assurer que le programme réalisé effectue correctement les tâches attendues. Vous l'avez fait jusqu'à présent en analysant les sorties des programmes, en utilisant `console.log` pour afficher également des résultats intermédiaires.

Cette approche trouve rapidement ses limites quand les projets grossissent et que plusieurs personnes sont impliquées². Il faut alors industrialiser le test. Il existe des tas d'approches, de boîtes à outils, d'environnement, mais un principe est de séparer le test du code utile. Concernant le test de petites fonctionnalités, "unitaires", un autre principe, respecté avec moins de vigueur que le précédent, est d'écrire les tests avant le code, ce qui a forgé le terme "Test Driven Development", TDD.

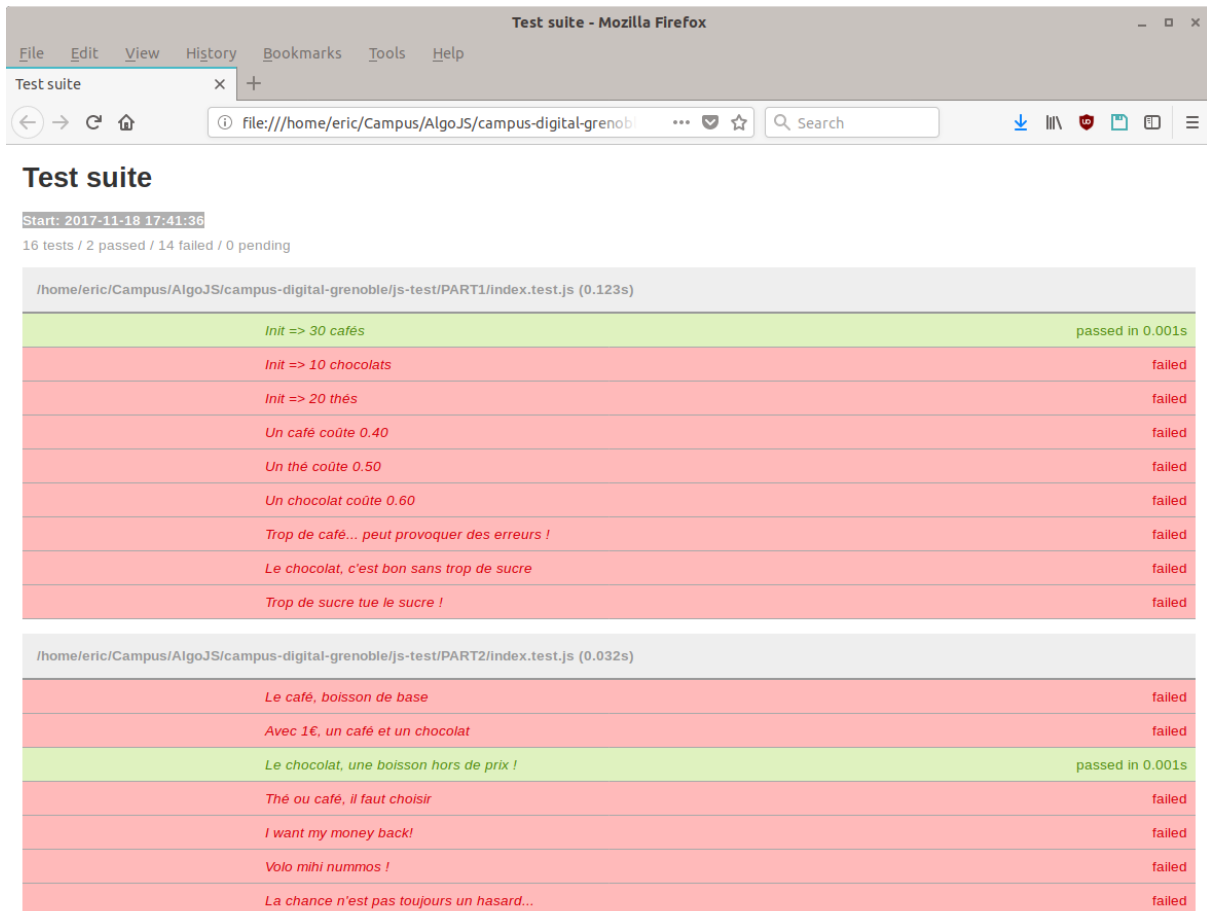
Ce projet est une initiation : les tests sont écrits (dans chaque répertoire PART1 et PART2, vous trouverez un fichier *index.test.js*) et le code (les fichiers *index.js*) doit être corrigé pour que les tests indiquent un fonctionnement correct.

¹ Un interpréteur (ou une machine) Javascript - vous avez utilisé un tel interpréteur dans une sandbox ou dans votre navigateur, mais il peut aussi être utilisé directement

² À ce propos, on peut considérer qu'un développeur et ce même développeur retrouvant son code quelques semaines après sont deux personnes différentes...

Le “framework” de test utilisé dans ce projet est Jest (<https://facebook.github.io/jest/> pour aller plus loin, mais l’objectif est d’intuiter ce qui se passe).

Pour lancer les tests, exécuter la commande “*npm start*” (allez voir dans le fichier *package.json* pour la définition de cette tâche). Cela devrait générer le fichier *test-report.html* et l’ouvrir dans votre navigateur :



Test suite	
Start: 2017-11-18 17:41:36	
16 tests / 2 passed / 14 failed / 0 pending	
/home/eric/Campus/AlgoJS/campus-digital-grenoble/js-test/PART1/index.test.js (0.123s)	
Init => 30 cafés	passed in 0.001s
Init => 10 chocolats	failed
Init => 20 thés	failed
Un café coûte 0.40	failed
Un thé coûte 0.50	failed
Un chocolat coûte 0.60	failed
Trop de café... peut provoquer des erreurs !	failed
Le chocolat, c'est bon sans trop de sucre	failed
Trop de sucre tue le sucre !	failed
/home/eric/Campus/AlgoJS/campus-digital-grenoble/js-test/PART2/index.test.js (0.032s)	
Le café, boisson de base	failed
Avec 1€, un café et un chocolat	failed
Le chocolat, une boisson hors de prix !	passed in 0.001s
Thé ou café, il faut choisir	failed
I want my money back!	failed
Volo mihi nummos !	failed
La chance n'est pas toujours un hasard...	failed

Chaque “paquet” correspond à PART1 ou PART2 (l’ordre n’est pas établi). Chaque ligne indique un test et s’il a réussi ou non.

Votre mission consiste pour chaque partie à ouvrir les fichiers de définition des test et le code testé et d’apporter les corrections nécessaires... en gardant bien entendu la fonctionnalité attendue !

Pour obtenir plus de détails de la raison d’un échec, vous pouvez consulter les informations produites sur la ligne de commande.