

Test en node.js

Mocha, Chai, Sinon

César Augusto Rodriguez

Qualité du logiciel et métriques

19 Septembre 2024

Ordre de cette présentation

- 1 Introduction
- 2 Test-Driven Development
- 3 Comment faire des tests ?
- 4 Mocha & Chai
- 5 Sinon
- 6 Conclusion

Introduction

Qu'est-ce que c'est ?

C'est le processus de vérification individuelle des modules et fonctions dans notre code pour s'assurer qu'ils fonctionnent comme prévu.

Pourquoi est-ce important ?

- Garantir le bon fonctionnement du code.
- Prévenir les bugs avant qu'ils ne se produisent.
- La base du Test-Driven Development (TDD).

Test-Driven Development

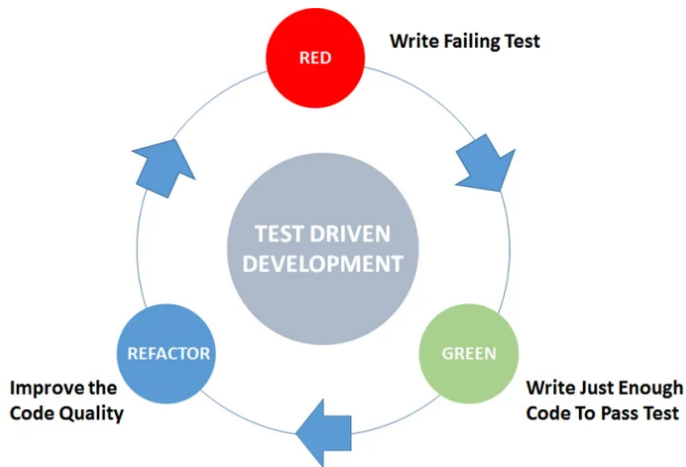


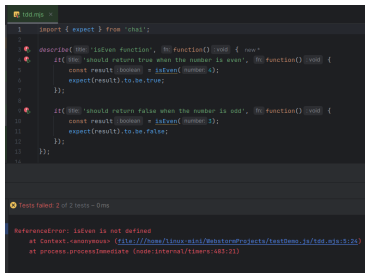
Figure: Étapes du TDD

Example TDD

Imaginons que nous voulons faire un programme simple qui vérifie si un nombre est pair ou non.

Première étape **Rouge**:

- Écrivons les tests d'abord sans même pas penser à notre fonction.
- Les tests vont échouer car notre fonction `isEven()` n'existe pas.



```
1 import { expect } from 'chai';

2 describe('isEven function', () => {
3   it('should return true when the number is even', () => {
4     const result = isEven(4);
5     expect(result).toBe(true);
6   });
7
8   it('should return false when the number is odd', () => {
9     const result = isEven(3);
10    expect(result).toBe(false);
11  });
12 });

13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Tests failed: 2 of 2 tests - 0ms

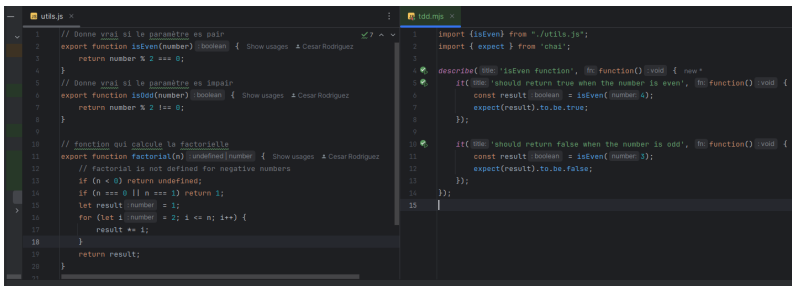
ReferenceError: isEven is not defined
at Context.<anonymous> (file:///home/linux/.vs/DevstoreProjects/testDemo.js/test_mis.3.24)
at process.processImmediate (node:internal/timers:463:21)

Figure: Red step

Example TDD

Deuxième étape Vert:

- On fait les modules nécessaires pour faire passer nos tests, sans aller plus loin. La fonction `isEven()` est faite sur `utils.js`



```
utils.js
1 // Donne vrai si le paramètre es pair
2 export function isEven(number) {
3   return number % 2 === 0;
4 }
5 // Donne vrai si le paramètre es impair
6 export function isOdd(number) {
7   return number % 2 !== 0;
8 }
9
10 // fonction qui calcule la factorielle
11 export function factorial(n) {
12   // factorial is not defined for negative numbers
13   if (n < 0) return undefined;
14   if (n === 0 || n === 1) return 1;
15   let result = 1;
16   for (let i = 2; i <= n; i++) {
17     result *= i;
18   }
19   return result;
20 }
21
```

```
utils.mjs
1 import { isEven } from './utils.js';
2 import { expect } from 'chai';
3
4 describe('isEven function', function() {
5   it('should return true when the number is even', function() {
6     const result = isEven(4);
7     expect(result).to.be.true;
8   });
9
10   it('should return false when the number is odd', function() {
11     const result = isEven(3);
12     expect(result).to.be.false;
13   });
14 });
15
```

Figure: Green step

- Les tests vont passer maintenant.

Example TDD

```
tdd.mjs
1  import {isEven} from './utils.js';
2  import { expect } from 'chai';
3
4  describe('isEven function', function() {
5    it('should return true when the number is even', function() {
6      const result = isEven(4);
7      expect(result).to.be.true;
8    });
9
10   it('should return false when the number is odd', function() {
11     const result = isEven(3);
12     expect(result).to.be.false;
13   });
14 });
15
```

```
Terminal
~/WebstormProjects/testDemo.js git:(master)
npx mocha tdd.mjs

isEven function
  ✓ should return true when the number is even
  ✓ should return false when the number is odd

2 passing (4ms)
```

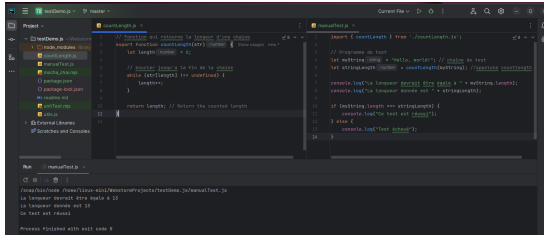
Troisième étape **Refactoring**:

- On améliore la qualité du code, et on peut refaire le cycle à nouveau.
- Dans ce cas simple on n'a pas besoin.

Comment faire des tests...

Manuellement

- Écrire nos propres fonctions.
- Utilisation personnalisée des fonctions d'affichage, ex. `console.log()`.
- Écrire du code spécifique pour chaque module.



The screenshot shows a code editor with two files: `countLength.js` and `manualTest.js`. The `countLength.js` file contains a function `countLength(str)` that returns the length of a string. The `manualTest.js` file contains a test program that imports the `countLength` function and tests it with various inputs. The test program uses `console.log` to output the results of the function calls. The output of the test is shown in the console at the bottom of the editor.

```
// countLength.js
// Fonction qui calcule la longueur d'une chaîne
export function countLength(str) {
  let length = str.length;
  return length;
}
```

```
// manualTest.js
import { countLength } from './countLength.js';

// Programme de test
let myString = 'Hello, world!'; // chaîne de test
let stringLength = countLength(myString); // appelle countLength

console.log('La longueur devrait être égale à ' + myString.length);
console.log('La longueur devrait être ' + stringLength);

if (myString.length === stringLength) {
  console.log('Le test est réussi');
} else {
  console.log('Test échoué');
}
```

```
Run: manualTest.js
/home/isaac/node /home/isaac/monstang/projects/textbook/js/manualTest.js
La longueur devrait être égale à 13
La longueur devrait être 13
Le test est réussi
Process finished with exit code 0
```

Figure: Exemple d'un test manuel

- **Avantages :**

- Fonctionne bien dans des cas particuliers.
- Peut être idéal là où les librairies ne sont pas disponibles.
 - Exemple : petits projets, impossibilité d'utiliser une librairie...
- Nécessite moins de connaissances sur des librairies externes.

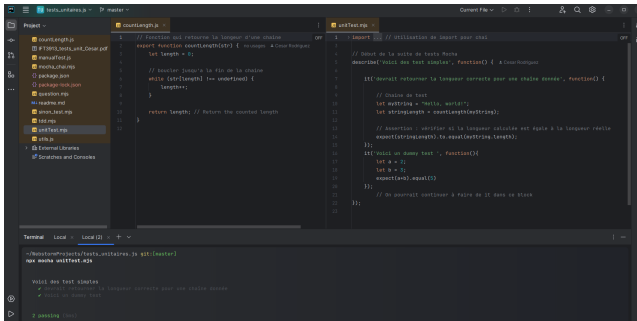
- **Inconvénients :**

- Risque de copier-coller, mauvais patrons de conception.
- Difficile à gérer avec un grand nombre de modules.

Type de tests

Tests unitaires avec des bibliothèques

- À l'aide de *frameworks* comme Mocha, Chai, Sinon, etc.
- Plus facile à lire et à entretenir.
- Open-source, maintenues par la communauté.



```
Project > tests-unitaires.js - master <
  countLength.js
  #1085_Mocha_Land_Osca.pdf
  sinonTest.js
  package.json
  package-lock.json
  question.mjs
  README.md
  sinon_test.mjs
  test.mjs
  unitTest.mjs
  utils.js
  External Libraries
  Scratches and Consoles

// Fonction qui retourne la longueur d'une chaîne
export function countLength(str) { // to-do : à compléter
  let length = 0;

  // Boucle jusqu'à la fin de la chaîne
  while (str[length] != undefined) {
    length++;
  }

  return length; // Return the counted length
}

// Utilisation de import pour chai
import { expect } from 'chai';

// Début de la suite de tests Mocha
describe('Voici des tests simples', function() {
  // Vérifier si la fonction retourne la longueur correcte pour une chaîne donnée, fonction() {
    // Chaîne de test
    let testString = 'Mocha, sinon!';
    let stringLength = countLength(testString);

    // Assertion : vérifier si la longueur calculée est égale à la longueur réelle
    expect(stringLength).to.equal(testString.length);
  });

  // Voici un dummy test -> fonction() {
    let a = 1;
    let b = 1;
    expect(a).equal(b);
  });

  // On pourrait continuer à écrire des it dans ce block
});
```

Figure: Demo simple plusieurs tests describe(), it()

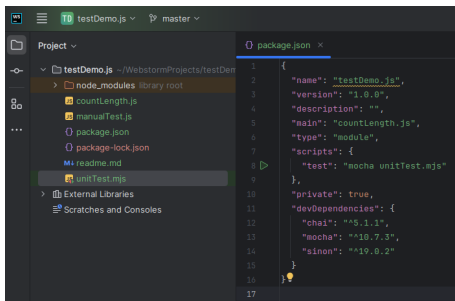
Frameworks in JavaScript

Mocha, Chai et Sinon (configuration)

Installation des bibliothèques

```
npm install mocha chai sinon --save-dev
```

Bien configurer votre fichier json avec "type": "module" et "test": "mocha nom_du_fichier"



The screenshot shows an IDE with a project named 'testDemo.js'. The file explorer on the left shows a directory structure with files like 'countLength.js', 'manualTest.js', 'package.json', 'package-lock.json', 'readme.md', and 'unitTest.mjs'. The 'package.json' file is open in the editor, showing the following configuration:

```
1 {
2   "name": "testDemo.js",
3   "version": "1.0.0",
4   "description": "",
5   "main": "countLength.js",
6   "type": "module",
7   "scripts": {
8     "test": "mocha unitTest.mjs"
9   },
10  "private": true,
11  "devDependencies": {
12    "chai": "^5.1.1",
13    "mocha": "^10.7.3",
14    "sinon": "^19.0.2"
15  }
16 }
```

Figure: Exemple de projet JSON

Mocha

- Possiblement la librairie la plus populaire de Node.js pour les test unitaires.
- Permet l'organisation de tests dans de blocs grâce aux fonctions `describe()`, `it()`.
- Souvent accompagné de Chai (prochaines slides).
- [Site officiel Mocha](#).
- Extension `.mjs`

Chai

- Permet l'utilisation des assertions sur les test.
 - Par exemple `assert.equal(factorial(5), 120); // 5! = 120`
- [Site officiel de Chai](#)

Avantages vs inconvénients

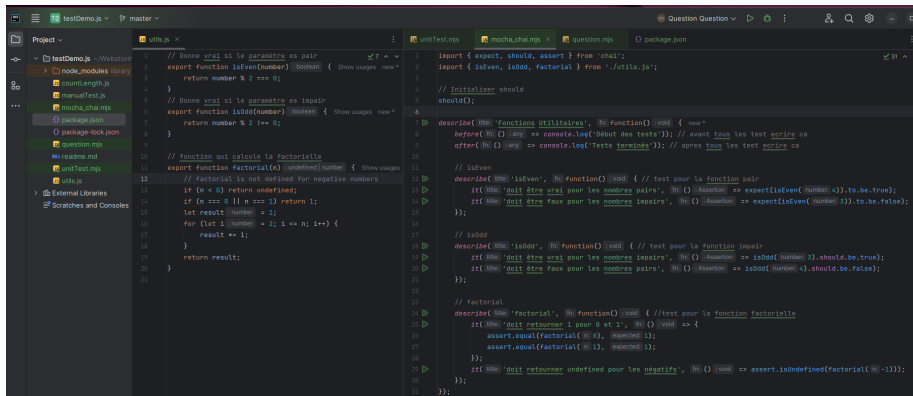
- **Avantages:**

- Permet de faire une suite de tests dans un seul bloque.
 - `describe()`, `it()`.
- Permettent de faire de tests selon les résultats attendus, ex:
 - `assert.equal()`, `assert.isNull()`, etc...
- Permet de compter le nombre d'appels à une fonction `spy()`.

- **Désavantages**

- Demande des connaissances sur ces librairies.
- **Maintenance**
 - Configuration initiale de la librairie,
 - Mise à jour de la librairie, etc

Démo Chai



The screenshot shows a code editor with two files open: `utils.js` and `unitTest.mjs`. The `utils.js` file contains a function `isEven` and a function `factorial`. The `unitTest.mjs` file contains Chai assertions for these functions.

```
// utils.js
1 // Donne vrai si le paramètre est pair
2 export function isEven(number) {
3   return number % 2 === 0;
4 }
5 // Donne vrai si le paramètre est impair
6 export function isOdd(number) {
7   return number % 2 !== 0;
8 }
9
10 // fonction qui calcule la factorielle
11 export function factorial(n) {
12   // factorial is not defined for negative numbers
13   if (n < 0) return undefined;
14   if (n === 0 || n === 1) return 1;
15   let result = number + 1;
16   for (let i = number + 2; i <= n; i++) {
17     result *= i;
18   }
19   return result;
20 }
21
```

```
// unitTest.mjs
1 import { expect, should, assert } from 'chai';
2 import { isEven, isOdd, factorial } from './utils.js';
3
4 // Initialiser should
5 should();
6
7 describe('Fonctions Utilitaires', () => {
8   // before
9   before(() => console.log('Début des tests')); // avant tous les test écrire ça
10  // after
11  after(() => console.log('Tests terminés')); // après tous les test écrire ça
12
13  // isEven
14  describe('isEven', () => {
15    it('doit être vrai pour les nombres pairs', () => {
16      expect(isEven(4)).to.be.true;
17    });
18    it('doit être faux pour les nombres impairs', () => {
19      expect(isEven(3)).to.be.false;
20    });
21  });
22
23  // isOdd
24  describe('isOdd', () => {
25    it('doit être vrai pour les nombres impairs', () => {
26      expect(isOdd(3)).to.be.true;
27    });
28    it('doit être faux pour les nombres pairs', () => {
29      expect(isOdd(4)).to.be.false;
30    });
31  });
32
33  // factorial
34  describe('factorial', () => {
35    it('doit retourner 1 pour 0 et 1', () => {
36      expect(factorial(0)).to.equal(1);
37      expect(factorial(1)).to.equal(1);
38    });
39    it('doit retourner undefined pour les négatifs', () => {
40      expect(isUndefined(factorial(-1))).to.be.true;
41    });
42  });
43
44  // factorial
45  describe('factorial', () => {
46    it('doit retourner 24 pour 4', () => {
47      expect(factorial(4)).to.equal(24);
48    });
49  });
50
51  // factorial
52  describe('factorial', () => {
53    it('doit retourner 120 pour 5', () => {
54      expect(factorial(5)).to.equal(120);
55    });
56  });
57
58  // factorial
59  describe('factorial', () => {
60    it('doit retourner 720 pour 6', () => {
61      expect(factorial(6)).to.equal(720);
62    });
63  });
64
65  // factorial
66  describe('factorial', () => {
67    it('doit retourner 5040 pour 7', () => {
68      expect(factorial(7)).to.equal(5040);
69    });
70  });
71
72  // factorial
73  describe('factorial', () => {
74    it('doit retourner 40320 pour 8', () => {
75      expect(factorial(8)).to.equal(40320);
76    });
77  });
78
79  // factorial
80  describe('factorial', () => {
81    it('doit retourner 362880 pour 9', () => {
82      expect(factorial(9)).to.equal(362880);
83    });
84  });
85
86  // factorial
87  describe('factorial', () => {
88    it('doit retourner 3628800 pour 10', () => {
89      expect(factorial(10)).to.equal(3628800);
90    });
91  });
92
93  // factorial
94  describe('factorial', () => {
95    it('doit retourner 36288000 pour 11', () => {
96      expect(factorial(11)).to.equal(36288000);
97    });
98  });
99
100  // factorial
101  describe('factorial', () => {
102    it('doit retourner 362880000 pour 12', () => {
103      expect(factorial(12)).to.equal(362880000);
104    });
105  });
106
107  // factorial
108  describe('factorial', () => {
109    it('doit retourner 3628800000 pour 13', () => {
110      expect(factorial(13)).to.equal(3628800000);
111    });
112  });
113
114  // factorial
115  describe('factorial', () => {
116    it('doit retourner 36288000000 pour 14', () => {
117      expect(factorial(14)).to.equal(36288000000);
118    });
119  });
120
121  // factorial
122  describe('factorial', () => {
123    it('doit retourner 362880000000 pour 15', () => {
124      expect(factorial(15)).to.equal(362880000000);
125    });
126  });
127
128  // factorial
129  describe('factorial', () => {
130    it('doit retourner 3628800000000 pour 16', () => {
131      expect(factorial(16)).to.equal(3628800000000);
132    });
133  });
134
135  // factorial
136  describe('factorial', () => {
137    it('doit retourner 36288000000000 pour 17', () => {
138      expect(factorial(17)).to.equal(36288000000000);
139    });
140  });
141
142  // factorial
143  describe('factorial', () => {
144    it('doit retourner 362880000000000 pour 18', () => {
145      expect(factorial(18)).to.equal(362880000000000);
146    });
147  });
148
149  // factorial
150  describe('factorial', () => {
151    it('doit retourner 3628800000000000 pour 19', () => {
152      expect(factorial(19)).to.equal(3628800000000000);
153    });
154  });
155
156  // factorial
157  describe('factorial', () => {
158    it('doit retourner 36288000000000000 pour 20', () => {
159      expect(factorial(20)).to.equal(36288000000000000);
160    });
161  });
162
163  // factorial
164  describe('factorial', () => {
165    it('doit retourner 362880000000000000 pour 21', () => {
166      expect(factorial(21)).to.equal(362880000000000000);
167    });
168  });
169
170  // factorial
171  describe('factorial', () => {
172    it('doit retourner 3628800000000000000 pour 22', () => {
173      expect(factorial(22)).to.equal(3628800000000000000);
174    });
175  });
176
177  // factorial
178  describe('factorial', () => {
179    it('doit retourner 36288000000000000000 pour 23', () => {
180      expect(factorial(23)).to.equal(36288000000000000000);
181    });
182  });
183
184  // factorial
185  describe('factorial', () => {
186    it('doit retourner 362880000000000000000 pour 24', () => {
187      expect(factorial(24)).to.equal(362880000000000000000);
188    });
189  });
190
191  // factorial
192  describe('factorial', () => {
193    it('doit retourner 3628800000000000000000 pour 25', () => {
194      expect(factorial(25)).to.equal(3628800000000000000000);
195    });
196  });
197
198  // factorial
199  describe('factorial', () => {
200    it('doit retourner 36288000000000000000000 pour 26', () => {
201      expect(factorial(26)).to.equal(36288000000000000000000);
202    });
203  });
204
205  // factorial
206  describe('factorial', () => {
207    it('doit retourner 362880000000000000000000 pour 27', () => {
208      expect(factorial(27)).to.equal(362880000000000000000000);
209    });
210  });
211
212  // factorial
213  describe('factorial', () => {
214    it('doit retourner 3628800000000000000000000 pour 28', () => {
215      expect(factorial(28)).to.equal(3628800000000000000000000);
216    });
217  });
218
219  // factorial
220  describe('factorial', () => {
221    it('doit retourner 36288000000000000000000000 pour 29', () => {
222      expect(factorial(29)).to.equal(36288000000000000000000000);
223    });
224  });
225
226  // factorial
227  describe('factorial', () => {
228    it('doit retourner 362880000000000000000000000 pour 30', () => {
229      expect(factorial(30)).to.equal(362880000000000000000000000);
230    });
231  });
232
233  // factorial
234  describe('factorial', () => {
235    it('doit retourner 3628800000000000000000000000 pour 31', () => {
236      expect(factorial(31)).to.equal(3628800000000000000000000000);
237    });
238  });
239
240  // factorial
241  describe('factorial', () => {
242    it('doit retourner 36288000000000000000000000000 pour 32', () => {
243      expect(factorial(32)).to.equal(36288000000000000000000000000);
244    });
245  });
246
247  // factorial
248  describe('factorial', () => {
249    it('doit retourner 362880000000000000000000000000 pour 33', () => {
250      expect(factorial(33)).to.equal(362880000000000000000000000000);
251    });
252  });
253
254  // factorial
255  describe('factorial', () => {
256    it('doit retourner 3628800000000000000000000000000 pour 34', () => {
257      expect(factorial(34)).to.equal(3628800000000000000000000000000);
258    });
259  });
260
261  // factorial
262  describe('factorial', () => {
263    it('doit retourner 36288000000000000000000000000000 pour 35', () => {
264      expect(factorial(35)).to.equal(36288000000000000000000000000000);
265    });
266  });
267
268  // factorial
269  describe('factorial', () => {
270    it('doit retourner 362880000000000000000000000000000 pour 36', () => {
271      expect(factorial(36)).to.equal(362880000000000000000000000000000);
272    });
273  });
274
275  // factorial
276  describe('factorial', () => {
277    it('doit retourner 3628800000000000000000000000000000 pour 37', () => {
278      expect(factorial(37)).to.equal(3628800000000000000000000000000000);
279    });
280  });
281
282  // factorial
283  describe('factorial', () => {
284    it('doit retourner 36288000000000000000000000000000000 pour 38', () => {
285      expect(factorial(38)).to.equal(36288000000000000000000000000000000);
286    });
287  });
288
289  // factorial
290  describe('factorial', () => {
291    it('doit retourner 362880000000000000000000000000000000 pour 39', () => {
292      expect(factorial(39)).to.equal(362880000000000000000000000000000000);
293    });
294  });
295
296  // factorial
297  describe('factorial', () => {
298    it('doit retourner 3628800000000000000000000000000000000 pour 40', () => {
299      expect(factorial(40)).to.equal(3628800000000000000000000000000000000);
300    });
301  });
302
303  // factorial
304  describe('factorial', () => {
305    it('doit retourner 36288000000000000000000000000000000000 pour 41', () => {
306      expect(factorial(41)).to.equal(36288000000000000000000000000000000000);
307    });
308  });
309
310  // factorial
311  describe('factorial', () => {
312    it('doit retourner 362880000000000000000000000000000000000 pour 42', () => {
313      expect(factorial(42)).to.equal(362880000000000000000000000000000000000);
314    });
315  });
316
317  // factorial
318  describe('factorial', () => {
319    it('doit retourner 3628800000000000000000000000000000000000 pour 43', () => {
320      expect(factorial(43)).to.equal(3628800000000000000000000000000000000000);
321    });
322  });
323
324  // factorial
325  describe('factorial', () => {
326    it('doit retourner 36288000000000000000000000000000000000000 pour 44', () => {
327      expect(factorial(44)).to.equal(36288000000000000000000000000000000000000);
328    });
329  });
330
331  // factorial
332  describe('factorial', () => {
333    it('doit retourner 362880000000000000000000000000000000000000 pour 45', () => {
334      expect(factorial(45)).to.equal(362880000000000000000000000000000000000000);
335    });
336  });
337
338  // factorial
339  describe('factorial', () => {
340    it('doit retourner 3628800000000000000000000000000000000000000 pour 46', () => {
341      expect(factorial(46)).to.equal(3628800000000000000000000000000000000000000);
342    });
343  });
344
345  // factorial
346  describe('factorial', () => {
347    it('doit retourner 36288000000000000000000000000000000000000000 pour 47', () => {
348      expect(factorial(47)).to.equal(36288000000000000000000000000000000000000000);
349    });
350  });
351
352  // factorial
353  describe('factorial', () => {
354    it('doit retourner 362880000000000000000000000000000000000000000 pour 48', () => {
355      expect(factorial(48)).to.equal(362880000000000000000000000000000000000000000);
356    });
357  });
358
359  // factorial
360  describe('factorial', () => {
361    it('doit retourner 3628800000000000000000000000000000000000000000 pour 49', () => {
362      expect(factorial(49)).to.equal(3628800000000000000000000000000000000000000000);
363    });
364  });
365
366  // factorial
367  describe('factorial', () => {
368    it('doit retourner 36288000000000000000000000000000000000000000000 pour 50', () => {
369      expect(factorial(50)).to.equal(36288000000000000000000000000000000000000000000);
370    });
371  });
372
373  // factorial
374  describe('factorial', () => {
375    it('doit retourner 362880000000000000000000000000000000000000000000 pour 51', () => {
376      expect(factorial(51)).to.equal(362880000000000000000000000000000000000000000000);
377    });
378  });
379
380  // factorial
381  describe('factorial', () => {
382    it('doit retourner 3628800000000000000000000000000000000000000000000 pour 52', () => {
383      expect(factorial(52)).to.equal(3628800000000000000000000000000000000000000000000);
384    });
385  });
386
387  // factorial
388  describe('factorial', () => {
389    it('doit retourner 36288000000000000000000000000000000000000000000000 pour 53', () => {
390      expect(factorial(53)).to.equal(36288000000000000000000000000000000000000000000000);
391    });
392  });
393
394  // factorial
395  describe('factorial', () => {
396    it('doit retourner 362880000000000000000000000000000000000000000000000 pour 54', () => {
397      expect(factorial(54)).to.equal(362880000000000000000000000000000000000000000000000);
398    });
399  });
400
401  // factorial
402  describe('factorial', () => {
403    it('doit retourner 3628800000000000000000000000000000000000000000000000 pour 55', () => {
404      expect(factorial(55)).to.equal(3628800000000000000000000000000000000000000000000000);
405    });
406  });
407
408  // factorial
409  describe('factorial', () => {
410    it('doit retourner 36288000000000000000000000000000000000000000000000000 pour 56', () => {
411      expect(factorial(56)).to.equal(36288000000000000000000000000000000000000000000000000);
412    });
413  });
414
415  // factorial
416  describe('factorial', () => {
417    it('doit retourner 362880000000000000000000000000000000000000000000000000 pour 57', () => {
418      expect(factorial(57)).to.equal(362880000000000000000000000000000000000000000000000000);
419    });
420  });
421
422  // factorial
423  describe('factorial', () => {
424    it('doit retourner 3628800000000000000000000000000000000000000000000000000 pour 58', () => {
425      expect(factorial(58)).to.equal(3628800000000000000000000000000000000000000000000000000);
426    });
427  });
428
429  // factorial
430  describe('factorial', () => {
431    it('doit retourner 36288000000000000000000000000000000000000000000000000000 pour 59', () => {
432      expect(factorial(59)).to.equal(36288000000000000000000000000000000000000000000000000000);
433    });
434  });
435
436  // factorial
437  describe('factorial', () => {
438    it('doit retourner 362880000000000000000000000000000000000000000000000000000 pour 60', () => {
439      expect(factorial(60)).to.equal(362880000000000000000000000000000000000000000000000000000);
440    });
441  });
442
443  // factorial
444  describe('factorial', () => {
445    it('doit retourner 3628800000000000000000000000000000000000000000000000000000 pour 61', () => {
446      expect(factorial(61)).to.equal(3628800000000000000000000000000000000000000000000000000000);
447    });
448  });
449
450  // factorial
451  describe('factorial', () => {
452    it('doit retourner 36288000000000000000000000000000000000000000000000000000000 pour 62', () => {
453      expect(factorial(62)).to.equal(36288000000000000000000000000000000000000000000000000000000);
454    });
455  });
456
457  // factorial
458  describe('factorial', () => {
459    it('doit retourner 362880000000000000000000000000000000000000000000000000000000 pour 63', () => {
460      expect(factorial(63)).to.equal(362880000000000000000000000000000000000000000000000000000000);
461    });
462  });
463
464  // factorial
465  describe('factorial', () => {
466    it('doit retourner 3628800000000000000000000000000000000000000000000000000000000 pour 64', () => {
467      expect(factorial(64)).to.equal(3628800000000000000000000000000000000000000000000000000000000);
468    });
469  });
470
471  // factorial
472  describe('factorial', () => {
473    it('doit retourner 36288000000000000000000000000000000000000000000000000000000000 pour 65', () => {
474      expect(factorial(65)).to.equal(36288000000000000000000000000000000000000000000000000000000000);
475    });
476  });
477
478  // factorial
479  describe('factorial', () => {
480    it('doit retourner 362880000000000000000000000000000000000000000000000000000000000 pour 66', () => {
481      expect(factorial(66)).to.equal(362880000000000000000000000000000000000000000000000000000000000);
482    });
483  });
484
485  // factorial
486  describe('factorial', () => {
487    it('doit retourner 3628800000000000000000000000000000000000000000000000000000000000 pour 67', () => {
488      expect(factorial(67)).to.equal(3628800000000000000000000000000000000000000000000000000000000000);
489    });
490  });
491
492  // factorial
493  describe('factorial', () => {
494    it('doit retourner 36288000000000000000000000000000000000000000000000000000000000000 pour 68', () => {
495      expect(factorial(68)).to.equal(36288000000000000000000000000000000000000000000000000000000000000);
496    });
497  });
498
499  // factorial
500  describe('factorial', () => {
501    it('doit retourner 362880000000000000000000000000000000000000000000000000000000000000 pour 69', () => {
502      expect(factorial(69)).to.equal(362880000000000000000000000000000000000000000000000000000000000000);
503    });
504  });
505
506  // factorial
507  describe('factorial', () => {
508    it('doit retourner 3628800000000000000000000000000000000000000000000000000000000000000 pour 70', () => {
509      expect(factorial(70)).to.equal(3628800000000000000000000000000000000000000000000000000000000000000);
510    });
511  });
512
513  // factorial
514  describe('factorial', () => {
515    it('doit retourner 36288000000000000000000000000000000000000000000000000000000000000000 pour 71', () => {
516      expect(factorial(71)).to.equal(36288000000000000000000000000000000000000000000000000000000000000000);
517    });
518  });
519
520  // factorial
521  describe('factorial', () => {
522    it('doit retourner 362880000000000000000000000000000000000000000000000000000000000000000 pour 72', () => {
523      expect(factorial(72)).to.equal(362880000000000000000000000000000000000000000000000000000000000000000);
524    });
525  });
526
527  // factorial
528  describe('factorial', () => {
529    it('doit retourner 3628800000000000000000000000000000000000000000000000000000000000000000 pour 73', () => {
530      expect(factorial(73)).to.equal(3628800000000000000000000000000000000000000000000000000000000000000000);
531    });
532  });
533
534  // factorial
535  describe('factorial', () => {
536    it('doit retourner 36288000000000000000000000000000000000000000000000000000000000000000000 pour 74', () => {
537      expect(factorial(74)).to.equal(36288000000000000000000000000000000000000000000000000000000000000000000);
538    });
539  });
540
541  // factorial
542  describe('factorial', () => {
543    it('doit retourner 362880000000000000000000000000000000000000000000000000000000000000000000 pour 75', () => {
544      expect(factorial(75)).to.equal(362880000000000000000000000000000000000000000000000000000000000000000000);
545    });
546  });
547
548  // factorial
549  describe('factorial', () => {
550    it('doit retourner 3628800000000000000000000000000000000000000000000000000000000000000000000 pour 76', () => {
551      expect(factorial(76)).to.equal(3628800000000000000000000000000000000000000000000000000000000000000000000);
552    });
553  });
554
555  // factorial
556  describe('factorial', () => {
557    it('doit retourner 36288000000000000000000000000000000000000000000000000000000000000000000000 pour 77', () => {
558      expect(factorial(77)).to.equal(36288000000000000000000000000000000000000000000000000000000000000000000000);
559    });
560  });
561
562  // factorial
563  describe('factorial', () => {
564    it('doit retourner 362880000000000000000000000000000000000000000000000000000000000000000000000 pour 78', () => {
565      expect(factorial(78)).to.equal(362880000000000000000000000000000000000000000000000000000000000000000000000);
566    });
567  });
568
569  // factorial
570  describe('factorial', () => {
571    it('doit retourner 3628800000000000000000000000000000000000000000000000000000000000000000000000 pour 79', () => {
572      expect(factorial(79)).to.equal(3628800000000000000000000000000000000000000000000000000000000000000000000000);
573    });
574  });
575
576  // factorial
577  describe('factorial', () => {
578    it
```

Librairie intéressant pour simuler le comportement des fonctions.

- `spy()` permet de compter le nombre d'appel sur une fonction, et quel type d'arguments ont été utilisés.
- `stub()` permet de simuler le résultat d'une fonction
 - Par exemple si on doit faire de test sur un module qui utilise une dépendence qu'on ne s'intéresse pas.
 - lorsqu'une fonction veut forcer un erreur sur une fonction por analyser son comportement.
- [Site Web officiel de Sinon](#)

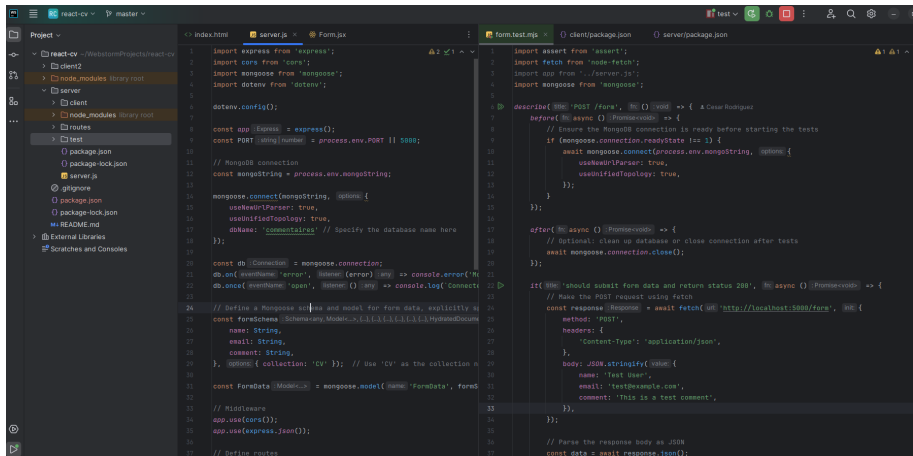
Demo Sinon

```
// utils.js
1 // Donne vrai si le paramètre es pair
2 export function isEven(number) {
3   return number % 2 === 0;
4 }
5 // Donne vrai si le paramètre es impair
6 export function isOdd(number) {
7   return number % 2 !== 0;
8 }
9 // fonction qui calcule la factorielle
10 export function factorial(n) {
11   // factorial is not defined for negative numbers
12   if (n < 0) return undefined;
13   if (n === 0 || n === 1) return 1;
14   let result = 1;
15   for (let i = 2; i <= n; i++) {
16     result *= i;
17   }
18   return result;
19 }

// sinon_test.mjs
1 import sinon from 'sinon';
2 import { isEven, isOdd } from './utils.js';
3 import { expect } from 'chai';
4
5 describe('Test avec Spy et Stub', () => {
6   it('espionne et compte les appels de isEven', () => {
7     const spy = sinon.spy(isEven);
8     spy(4); spy(6); // isEven est appelé deux fois avec paramètres 4 et 6.
9
10    expect(spy.calledTwice).to.be.true; // est-ce que le espion est appelé deux fois?
11    expect(spy.calledWith(4)).to.be.true; // est-ce qu'il a été appelé avec 4
12    expect(spy.calledWith(6)).to.be.true; // est-ce qu'il a été appelé avec 6
13  });
14
15   it('stub isEven pour toujours retourner vrai', () => {
16     const stub = sinon.stub().returns(true); // on set stub to return vrai toujours
17     expect(stub(5)).to.be.true; //return toujours vrai
18     expect(stub.calledOnce).to.be.true; // est-ce que le stub est appelé au moins une fois
19   });
20
21   it('stub isOdd pour toujours retourner faux', () => {
22     const stub = sinon.stub().returns(false);
23     expect(stub(7)).to.be.false;
24     expect(stub.calledOnce).to.be.true;
25   });
26 });
```

Figure: Exemple Sinon

Projet personnel Demo



```
Project >
├── react-cv ~/WebstormProjects/react-cv
├── client2
├── node_modules library root
├── server
│   ├── client
│   ├── routes
│   ├── node_modules library root
│   ├── package.json
│   ├── package-lock.json
│   ├── server.js
│   ├── .gitignore
│   ├── package.json
│   ├── package-lock.json
│   └── README.md
├── External Libraries
└── Scratches and Consoles

server.js
1 import express from 'express';
2 import cors from 'cors';
3 import mongoose from 'mongoose';
4 import dotenv from 'dotenv';
5
6 dotenv.config();
7
8 const app = express();
9 const PORT = (String) process.env.PORT || 5000;
10
11 // MongoDB connection
12 const mongoString = process.env.mongoString;
13
14 mongoose.connect(mongoString, {
15   useNewUrlParser: true,
16   useUnifiedTopology: true,
17   dbName: 'commentaires' // Specify the database name here
18 });
19
20 const db = mongoose.connection;
21 db.on('error', (error) => console.error('MongoDB connection error: ', error));
22 db.once('open', () => console.log('Connected to MongoDB'));
23
24 // Define a Mongoose schema and model for form data, explicitly as
25 const formSchema = Schema({
26   name: String,
27   email: String,
28   comment: String,
29 }, { collection: 'CV' }); // Use 'CV' as the collection name
30
31 const FormData = mongoose.model('FormData', formSchema);
32
33 // Middleware
34 app.use(cors());
35 app.use(express.json());
36
37 // Define routes
38
form.test.mjs
1 import assert from 'assert';
2 import fetch from 'node-fetch';
3 import app from '../server.js';
4 import mongoose from 'mongoose';
5
6 describe('POST /form', () => {
7   beforeEach(async () => {
8     // Ensure the MongoDB connection is ready before starting the tests
9     if (mongoose.connection.readyState !== 1) {
10       await mongoose.connect(process.env.mongoString, {
11         useNewUrlParser: true,
12         useUnifiedTopology: true,
13       });
14     }
15   });
16
17   after(async () => {
18     // Optional: clean up database or close connection after tests
19     await mongoose.connection.close();
20   });
21
22   it('should submit form data and return status 200', async () => {
23     // Make the POST request using fetch
24     const response = await fetch('http://localhost:5000/form', {
25       method: 'POST',
26       headers: {
27         'Content-Type': 'application/json',
28       },
29       body: JSON.stringify({
30         name: 'Test User',
31         email: 'test@example.com',
32         comment: 'This is a test comment',
33       }),
34     });
35
36     // Parse the response body as JSON
37     const data = await response.json();
38   });
39 });
```

Figure: Projet personnel

Conclusion

- Les test unitaires sont la base de la conception TDD.
- Les test unitaires servent à vérifier la bonne qualité du code et à prévenir des bugs.
- Dans javascript, Mocha, Chai et Sinon sont des librairies assez bonnes pour une telle conception.

Bibliographie

- [Udemy Node.js](#)
- [Développement TDD](#)
- [Chai](#)
- [Mocha](#)
- [Sinon](#)