

# Créer des tests unitaires JS avec Jest

<https://jestjs.io/fr/>



## Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

# Introduction

## De la touche "F5" aux frameworks de tests

Les tests unitaires sont aujourd'hui une norme dans le développement des applications. L'amplification des techniques Agiles et du mouvement [Software Craftsmanship](#) ont poussé à mettre les tests unitaires comme prérequis au développement d'applications.

Concernant le développement d'applications front en javascript, les tests se limitent souvent à une vérification manuelle du comportement attendu sur le navigateur. La touche "F5", les "alert" ou « console.log » en sont les principaux outils. Qui n'a pas passé des heures à rafraîchir sa page et à regarder les messages d'alertes pour comprendre le changement de comportement d'un module lors d'une modification du code ?

Ceci entraîne une perte de temps énorme, la confiance envers son code lorgnant vers 0 ainsi qu'une maintenance qui devient quasiment impossible.

Heureusement, au fil du temps les navigateurs ont sorti des outils permettant de faciliter les tests du code tels que Firebug pour Firefox ou le puissant outil de développement présent dans Chrome. Ces outils sont d'une aide inestimable mais les tests sont toujours faits manuellement.

Des frameworks de tests unitaires pour Javascript existent pourtant depuis plus de 10 ans. L'apparition de [Node.js](#) et de frameworks de développement front en javascript ([Angular](#), [Backbone](#), [Ember](#), [React](#), [VueJS](#)...) ont fait passer le javascript dans une dimension supérieure et permettent aujourd'hui le développement partiel ou total d'applications dans ce langage. Ce développement impose de facto l'utilisation de tests (unitaires, d'intégration...).

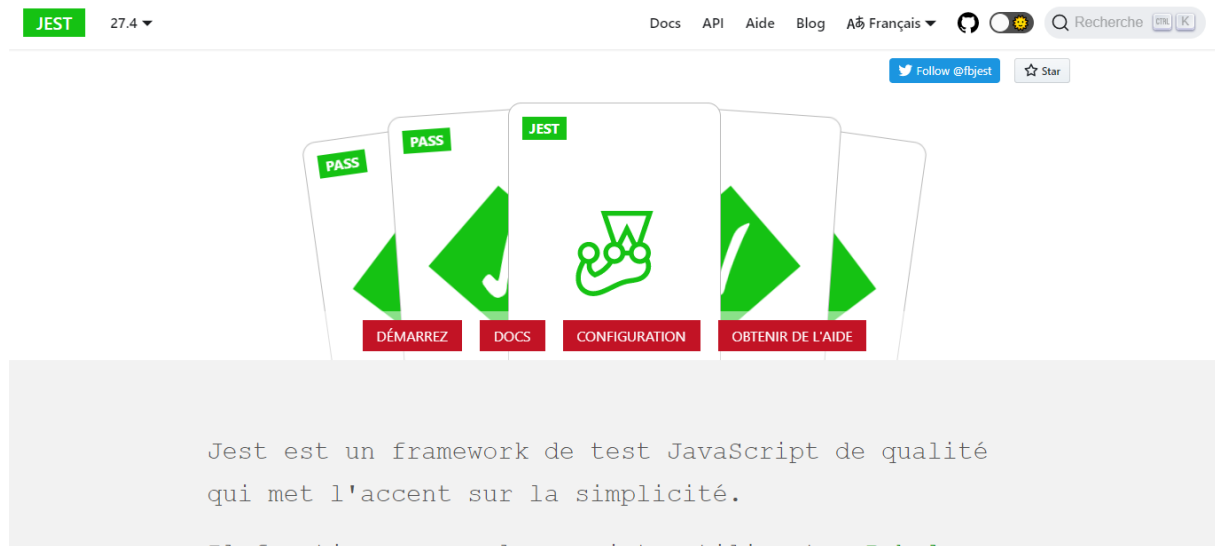
Des frameworks de tests automatisés tel que [Jasmine](#), [Mocha](#), [QUnit](#) ou [Jest](#), nous permettent de nous assurer que nos développement répondent aux exigences spécifiées.

### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité  
**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

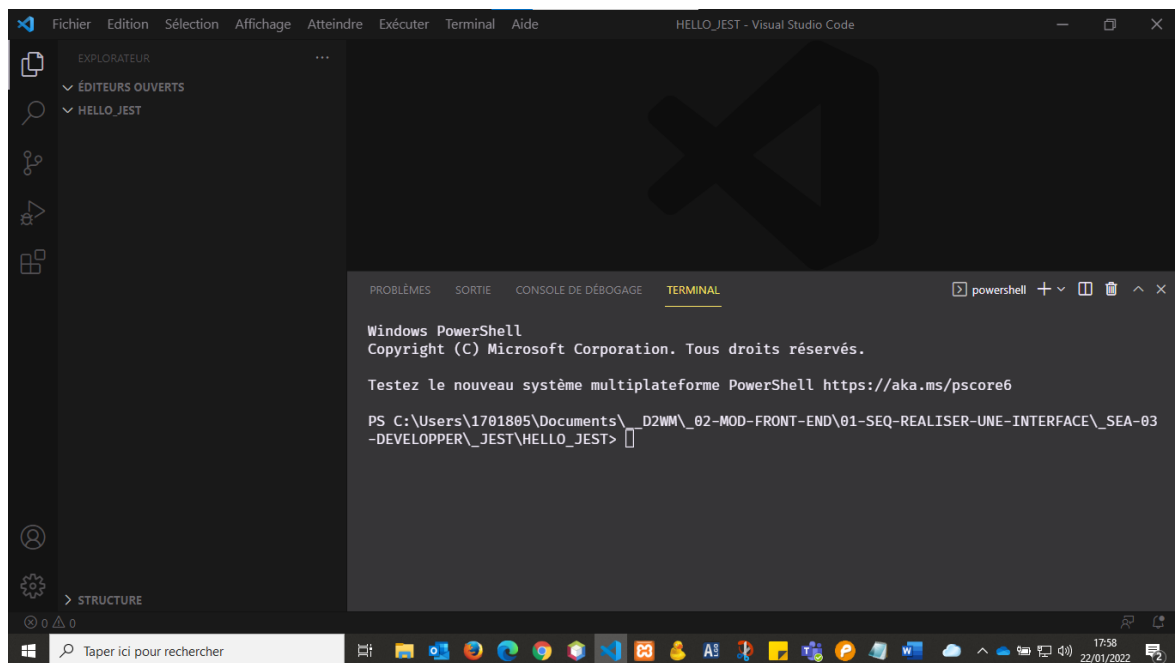
# Installer



*Cliquez sur le bouton « DEMARREZ »*

Sur votre disque dur créer un répertoire intitulé « HELLO\_JEST »

Ouvrez le dans VS Code puis accédez au terminal



## Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

Tapez la commande `npm init -y` pour créer un fichier ***package.json***

*Le -y permet de répondre automatiquement "oui" à toutes les invites que npm pourrait demander sur la ligne de commande.*

```

Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell https://aka.ms/pscore6

PS C:\Users\1701805\Documents\__D2WM\__02-MOD-FRONT-END\01-SEQ-REALISER-UNE-INTERFACE\_SEA-03-DEVELOPPER\_JEST\HELLO_JEST> npm init -y
Wrote to C:\Users\1701805\Documents\__D2WM\__02-MOD-FRONT-END\01-SEQ-REALISER-UNE-INTERFACE\_SEA-03-DEVELOPPER\_JEST\HELLO_JEST\package.json:

{
  "name": "HELLO_JEST",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

```

Installez Jest en suivant les indications de la doc, dans le terminal :

```
npm install --save-dev jest
```

*-dev précise qu'il s'agit d'un outil de développement*

```
-DEVELOPPER\JEST\HELLO_JEST> npm install --save-dev jest
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@^2.3.2 (node_modules\jest-haste-map\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm WARN HELLO_JEST@1.0.0 No description
npm WARN HELLO_JEST@1.0.0 No repository field.

+ jest@27.4.7
added 323 packages from 263 contributors and audited 324 packages in 20.158s

26 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

## Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

Vous devriez avoir quelque chose comme l'écran ci-dessus.

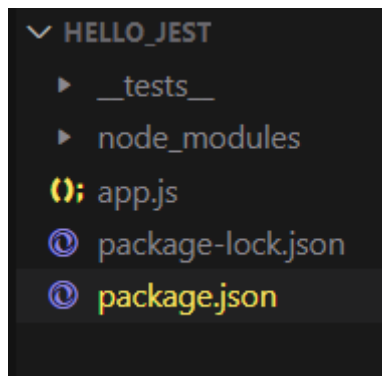
Si vous ouvrez le fichier package.json, vous devez voir la version de Jest installée

```
package.json X
package.json > ...
1 {
2   "name": "hello_jest",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "devDependencies": {
13    "jest": "^27.4.7"
14  }
15 }
16
```

Ici la 27.4.7

## Démarrer

Créer dans votre répertoire **HELLO\_JEST** un fichier **app.js** et un répertoire de test nommé **\_\_tests\_\_**



**app.js** contiendra le code de l'application

**\_\_tests\_\_** contiendra les fichiers de test

Dans le fichier app.js :

### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

```
(); app.js ×
() app.js > [?] <unknown>
1  const sentence = "Hello World";
2
3  // On exporte la variable que l'on doit tester
4  module.exports = {
5      sentence
6  }
```

Dans le répertoire `__tests__`, créer un fichier intitulé **app.test.js**

Par convention, on nomme le fichier de test du nom du fichier à tester avec `.test`, exemple ici avec `app.js` et `app.test.js`

Dans le fichier **app.test.js** :

Il faut récupérer l'objet à tester

```
(); app.test.js × (); app.js
__tests__ > () app.test.js > [?] app
1  // Récupération de l'objet à tester
2  const app = require('../app.js')
```

La directive "**require**" indique à JavaScript d'importer la totalité du module demandée. Attention, si le module en question est lourd, cela peut allonger le délai d'affichage d'une page.

Ensuite il faut décrire l'objet à tester :

```
(); app.test.js × (); app.js
__tests__ > () app.test.js > ...
1  // Récupération de l'objet à tester
2  const app = require('../app.js')
3
4  // Décrire le test
5  describe('nomDuTest', () => {
6      it('should retrieve a sentence', () => {
7          expect(app.sentence.length).toBeGreaterThan(0);
8      });
9  });
```

#### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité  
**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

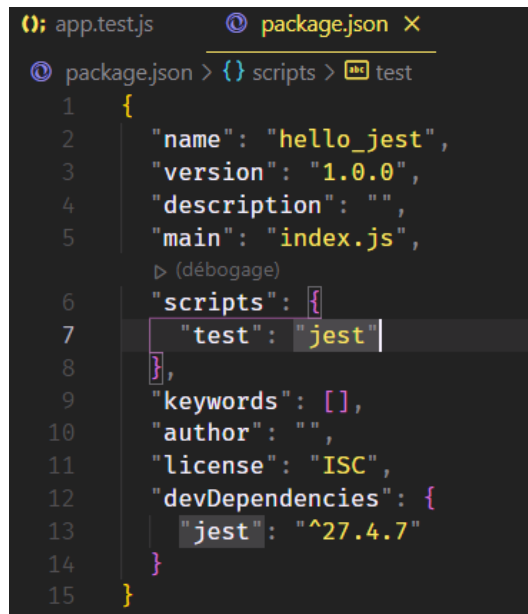
Réalisation Stéphane Pontonnier – 2021

**Describe** : <https://jestjs.io/fr/docs/api#describename-fn>

**it** : décrit ce que fait le test et appelle une fonction de callback

**expect** : On décrit ce à quoi on s'attend (dans l'exemple ci-dessus, une phrase plus grande que 0)

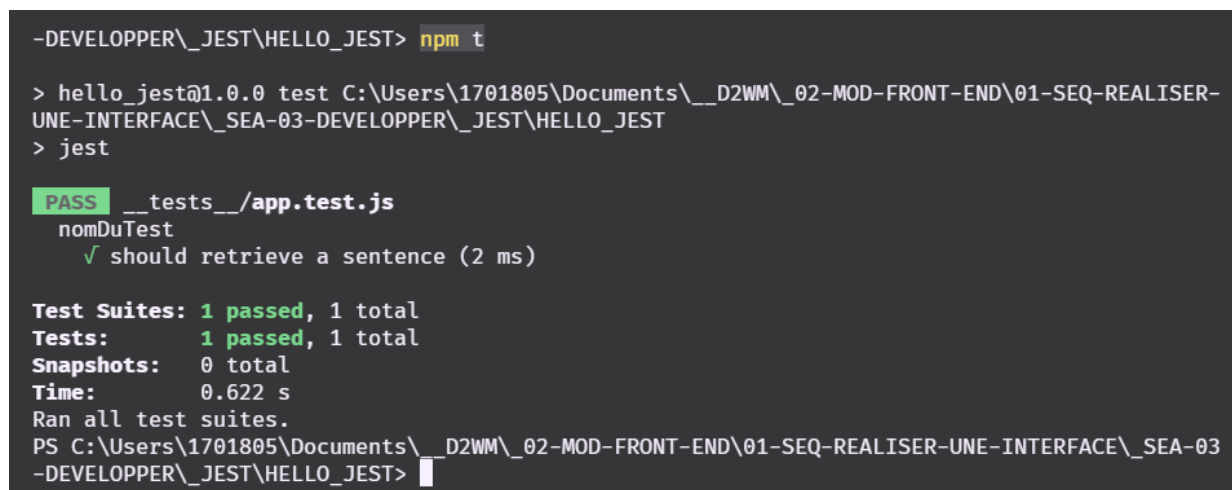
Pour tester votre code, dans le fichier **package.json** :



```
0: app.test.js  package.json X
package.json > {} scripts > test
1  {
2    "name": "hello_jest",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "jest"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "devDependencies": {
13     "jest": "^27.4.7"
14   }
15 }
```

Modifier la valeur de « test »

Dans la console de VS Code tapez **npm run test** ou en abrégé **npm t** :



```
-DEVELOPPER\_JEST\HELLO_JEST> npm t
> hello_jest@1.0.0 test C:\Users\1701805\Documents\_D2WM\_02-MOD-FRONT-END\01-SEQ-REALISER-
UNE-INTERFACE\_SEA-03-DEVELOPPER\_JEST\HELLO_JEST
> jest

PASS  __tests__/app.test.js
  nomDuTest
    ✓ should retrieve a sentence (2 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.622 s
Ran all test suites.
PS C:\Users\1701805\Documents\_D2WM\_02-MOD-FRONT-END\01-SEQ-REALISER-UNE-INTERFACE\_SEA-03-DEVELOPPER\_JEST\HELLO_JEST>
```

Le test nommé **nomDuTest** et qui doit retourner une phrase a été réalisé sans erreurs. Cependant, il est primordial de tester que le test échoue également.

#### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

Reprenez le fichier **app.js** et modifiez-le :

```
() app.js > [?] sentence
1  const sentence = "";
2
3  // On exporte la variable que l'on doit tester
4  module.exports = {
5    sentence
6  }
```

Dans la console, tapez **npm t** :

```
FAIL __tests__/app.test.js
  nomDuTest
    ✕ should retrieve a sentence (3 ms)

● nomDuTest › should retrieve a sentence

  expect(received).toBeGreaterThan(expected)

    Expected: > 0
    Received: 0
```

Le résultat attendu devait être > 0 (Expected) et ce qui a été reçu est 0 (Received)

Afin de bénéficier de plus de confort, installez le package **@types/jest**

Dans la console de VS Code tapez **npm i -D @types/jest**

```
PS C:\Users\1701805\Documents\_D2WM\_02-MOD-FRONT-END\01-SEQ-REALISER-UNE-INTERFACE\_SEA-03-DEVELOPPER\_JEST\HELLO_JEST> npm i -D @types/jest
npm WARN hello_jest@1.0.0 No description
npm WARN hello_jest@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ @types/jest@27.4.0
added 1 package from 27 contributors and audited 325 packages in 2.452s

26 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Allez dans **app.test.js** et créez un nouveau test :

#### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021



```

4 // Décrire le test
5 describe('nomDuTest', () => {
6   it('should retrieve a sentence', () => {
7     expect(app.sentence.length).toBeGreaterThan(0);
8   });
9   it('should have a length of 11 letters', () => {
10    expect(app.sentence.length).toEqual(11);
11  });
12 });

```

Dans **app.js** modifier la valeur de la constante à « Hello World »

Dans la console testez à nouveau avec **npm t**

```

PASS  __tests__/app.test.js
  nomDuTest
    ✓ should retrieve a sentence (2 ms)
    ✓ should have a length of 11 letters (1 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.481 s, estimated 1 s
Ran all test suites.

```

On peut voir une test suite définie par describe composée de 2 tests qui passent. Changer la valeur 11 par 10 dans **app.test.js**, lancez le test et regardez la sortie dans la console.

Pour ne plus avoir à lancer **npm run test** ou **npm t** à chaque fois, dans le fichier **package.json**

```

6   "scripts": {
7     "test": "jest --watchAll --verbose"
8   },

```

**--watchAll** : Surveille les changements dans les fichiers et réexécute tous les tests lorsque quelque chose change

**--verbose** : Affiche les résultats de chaque test avec la hiérarchie de la suite de test.

Si vous tapez **w** dans la console, vous avez accès aux autres commandes du **watch mode**

#### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

### Watch Usage

- › Press f to run only failed tests.
- › Press o to only run tests related to changed files.
- › Press p to filter by a filename regex pattern.
- › Press t to filter by a test name regex pattern.
- › Press q to quit watch mode.
- › Press Enter to trigger a test run.

## TDD

**Test-Driven Development (TDD)**, ou **Développements Pilotés par les Tests** en français, est une méthode de développement de logiciel, qui consiste à concevoir un logiciel par petits pas, de façon itérative et incrémentale, en écrivant chaque test avant d'écrire le code source et en remaniant le code continuellement. (Source [Wikipédia](#)).

Atelier 1 : Développer des tests pour savoir si un mot est un palindrome

Dans le fichier **app.test.js**, supprimer les tests précédents :

```
❯; app.test.js × ❯; app.js  ⓘ package.json 1
__tests__ > ❯; app.test.js > ...
1 // Récupération de l'objet à tester
2 const app = require('../app.js')
3
4 // Décrire le test
5 describe('Palindrom', () => {
6
7 });
```

Et créer un nouveau test

### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

```
// Décrire le test palindrom
describe('Palindrom', () => {
  it('should be a palindrom', ()=>{
    expect(app.isPalindrom('kayak')).toEqual(true);
  });
});
```

Si vous lancez le test via la console, il échoue, **isPalindrom is not a function**

```
FAIL __tests__/app.test.js
  Palindrom
    × should be a palindrom (3 ms)

● Palindrom › should be a palindrom

TypeError: app.isPalindrom is not a function
```

Dans le fichier **app.js**, il faut donc écrire la fonction isPalindrome

```
() app.js > <unknown>
1  function isPalindrom(word){
2    return true;
3  }
4  module.exports = {
5    isPalindrom
6  }
```

Le test n'échoue plus dans la console, mais cette fonction ne sert à rien. Ecrivez l'algorithme qui permet de tester si un mot est un palindrome (split(), join(), etc...)

#### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

```

0; app.js > isPalindrom
1  function isPalindrom(word){
2      // tester si word est un palindrome
3      // retourner true si ok
4  }
5  module.exports = {
6      isPalindrom
7  }

```

Ensuite, dans app.test.js, écrivez le test qui vérifie qu'un mot n'est pas un palindrome

```

__tests__ > 0; app.test.js > ...
1  // Récupération de l'objet à tester
2  const app = require(' ../app.js')
3
4  // Décrire le test palindrom
5  describe('Palindrom', () => {
6      // test mot est un palindrome
7      it('should be a palindrom', ()=>{
8          expect(app.isPalindrom('kayak')).toEqual(true);
9      });
10     // test mot n'est pas un palindrome
11     // instructions
12 });

```

## Atelier 2 : Création d'un module de modération de forum

Dans le fichier **app.test.js**, écrivez le test de modération. On teste que notre module app contienne une fonction **containForbiddenWords** qui nous indique si une chaîne de caractères possède des termes interdits.

### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

```

tests_ > 0; app.test.js > ...
1 // Récupération de l'objet à tester
2 const app = require('../app.js')
3
4 // Décrire le test palindrom
5 > describe('Palindrom', () => {
13 });
14
15 describe('Moderator', () => {
16   it('contient des mots interdits', () => {
17     expect(app.containForbiddenWords('Vous êtes des nuls')).toEqual(true);
18   });
19 });

```

Lancez le test via `npm t`

```

● Moderator › contient des mots interdits

expect(received).toEqual(expected) // deep equality

Expected: true
Received: undefined

```

Le test naturellement échoue, dans app.js il faut maintenant coder la fonction **containForbiddenWords** qui va renvoyer **true** si un (ou des) mot(s) sont dans le message.

```

0; app.js > containForbiddenWords
1 > function isPalindrom(word){
5 }
6
7 // liste de mots interdits
8 const forbiddenWords = ['nuls', 'nul', 'imbécile', 'imbéciles', 'idiots', 'idiot', 'débiles', 'débile'];
9
10 function containForbiddenWords(message){
11   // instructions
12   // instructions
13   // instructions
14   // instructions
15 }
16
17
18 module.exports = {
19   isPalindrom,
20   containForbiddenWords
21 }

```

Commencez par créer un tableau qui contient des mots interdits. Ensuite codez les instructions de la fonction. Pensez bien à exporter votre fonction pour pouvoir l'utiliser dans les tests (cf ligne 20 de l'exemple ci-dessus).

Résultat :

#### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité  
**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

```
// liste de mots interdits
const forbiddenWords = ['nuls', 'nul', 'imbécile', 'imbéciles', 'idiots', 'idiot', 'débiles', 'débile'];

function containForbiddenWords(message){
  const msg = message.split(' ');
  let result = [];
  for (let index = 0; index < msg.length; index++) {
    const item = msg[index];
    if (forbiddenWords.includes(item)) {
      result.push(item);
    }
  }
  // Si le tableau result est > 0 donc a au moins une cellule de compléter
  if(result.length > 0){
    return true;
  }
  return false;
}
}
```

Votre fonction une fois écrite et fonctionnelle, lancez le test :

```
PASS __tests__/app.test.js
PASS __tests__/app.test.js
PASS __tests__/app.test.js
PASS __tests__/app.test.js
Palindrom
  ✓ should be a palindrom (1 ms)
Moderator
  ✓ contient des mots interdits

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.276 s, estimated 1 s
Ran all test suites.

Watch Usage: Press w to show more.
```

Testez également en situation d'échec :

```
● Moderator > contient des mots interdits

expect(received).toEqual(expected) // deep equality

Expected: true
Received: false

   15 | describe('Moderator', () =>{
   16 |   it('contient des mots interdits', () => {
>  17 |     expect(app.containForbiddenWords('Vous êtes des nulles')).toEqual(true);
      |                                                                    ^
   18 |   });
   19 | });
```

#### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité  
**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

Ici, le message de la ligne 17 ne contient pas de mots interdits, Expected est à true et Received à false donc le test est un échec.

Si vous avez besoin de lister rapidement des tests à faire, vous pouvez utiliser la syntaxe suivante :

```
describe('Moderator', () =>{
  it('contient des mots interdits', () => {
    expect(app.containForbiddenWords('Vous êtes des nuls')).toEqual(true);
  });
  it.todo('test suivant à réaliser');
  it.todo('test à réaliser plus tard');
});
```

Utilisez **it.todo** lorsque vous prévoyez d'écrire des tests. Ces tests seront mis en évidence dans le résumé de la sortie à la fin, afin que vous sachiez combien de tests il vous reste à faire.

```
Test Suites: 1 passed, 1 total
Tests:       2 todo, 2 passed, 4 total
Snapshots:   0 total
Time:        0.232 s, estimated 1 s
Ran all test suites.
```

**Remarque :** Si vous fournissez une callback de test alors le **it.todo** lancera une erreur.

Maintenant, dans **app.test.js**, écrivez un test qui remplace les mots interdits par xxx

```
describe('Moderator', () =>{
  it('contient des mots interdits', () => {
    expect(app.containForbiddenWords('Vous êtes des nuls')).toEqual(true);
  });
  it('remplacer les mots interdits par xxx', () => {
    // expect(app.)
  });
});
```

Le test :

#### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité  
**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

```
describe('Moderator', () => {
  it('contient des mots interdits', () => {
    expect(app.containForbiddenWords('Vous êtes des nuls')).toEqual(true);
  });
  it('remplacer les mots interdits par xxx', () => {
    expect(app.changeForbiddenWords("vous êtes nuls")).toEqual("vous êtes xxx");
  });
});
```

Evidemment, si on regarde la console, le test a échoué :

```
✓ contient des mots interdits
× remplacer les mots interdits par xxx (1 ms)

● Moderator › remplacer les mots interdits par xxx

TypeError: app.changeForbiddenWords is not a function
```

Il faut dans **app.js** écrire la fonction qui change les mots interdits par xxx

```
function changeForbiddenWords(msg){
  return 'vous êtes des xxx';
}

module.exports = {
  ...
  isPalindrom,
  containForbiddenWords,
  changeForbiddenWords
}
```

Le résultat :

#### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité  
**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021



```

function changeForbiddenWords(msg){
  // on récupère le résultat de la fonction précédente
  const hasForbiddenWords = containForbiddenWords(msg);
  // Si tout est ok on retourne la chaîne
  if (!hasForbiddenWords){
    return msg;
  }
  // Dans le cas où on doit changer qqch
  const words = msg.split(' ');
  //console.log('words', words);
  const result = words.map(w => {
    // si le tableau des termes interdits contient le terme de words
    if (forbiddenWords.includes(w)) {
      return 'xxx';
    } else {
      return w;
    }
  });
  //console.log('result', result);
  return result.join(' ');
}

```

La méthode **map()** crée un nouveau tableau avec les résultats de l'appel d'une fonction fournie sur chaque élément du tableau appelant. Ici, chaque mot interdit est remplacé par xxx.

Le résultat des tests

```

Moderator
  ✓ contient des mots interdits (1 ms)
  ✓ remplacer les mots interdits par xxx (2 ms)
  ✓ retourne la même phrase si pas de mots interdits détectés

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        0.287 s, estimated 1 s
Ran all test suites.

```

Maintenant dans **app.test.js**, écrivez le test qui vérifie que si la phrase testée ne contient aucun mot interdit, elle est retournée telle quelle

#### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité  
**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

```
describe('Moderator', () =>{
  it('contient des mots interdits', () => {
    expect(app.containForbiddenWords('Vous êtes des nuls')).toEqual(true);
  });
  it('remplacer les mots interdits par xxx', () => {
    expect(app.changeForbiddenWords("vous êtes des nuls")).toEqual("vous êtes des xxx");
  });
  it('retourne la même phrase si pas de mot interdit détecté', ()=>{
    expect(app.changeForbiddenWords("vous êtes de merveilleux stagiaires")).toEqual("vous êtes de merveilleux stagiaires");
  });
});
```

La suite de tests « Moderator » est presque complète, il est possible d'ajouter un test qui vérifie la casse de la phrase.

```
it('détecte si la casse est modifiée', () => {
  expect(app.changeForbiddenWords("vous êtes des imBécILES")).toEqual("vous êtes des xxx");
})
```

Vous devez modifier votre code dans **app.js**

```
for (let index = 0; index < msg.length; index++) {
  const item = msg[index].toLocaleLowerCase();
  if (forbiddenWords.includes(item)) {
    result.push(item);
  }
}
```

```
const result = words.map(w => {
  // si le tableau des termes interdits contient le terme de words
  if (forbiddenWords.includes(w.toLocaleLowerCase())) {
    return 'xxx';
  } else {
    return w;
  }
});
```

La méthode **toLocaleLowerCase()** renvoie la chaîne de caractères qui appelle la méthode en une chaîne de caractères représentées en minuscules, en tenant compte des correspondances de caractères propres aux différentes locales.

#### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité  
**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

# Utiliser Jest avec la syntaxe des ECMAScript modules (import et export default) plutôt qu'avec CommonJS

<https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Modules>

<https://tech-wiki.online/fr/commonjs.html>

<https://jestjs.io/fr/docs/ecmascript-modules>

<https://babeljs.io/>

Afin d'utiliser la syntaxe propre aux ECMAScript modules (avec les mots clés "export" et "import") plutôt que la syntaxe par défaut de CommonJS (avec "module.exports" et "require"), il faut installer quelques plugins Babel :

**npm i -D babel-jest @babel/core @babel/preset-env**

```
PS C:\Users\1701805\Documents\_D2WM\_02-MOD-FRONT-END\01-SEQ-REALISER-UNE-INTERFACE\_SEA-03-DEVELOPPER\_JEST\HELLO_JEST> npm i -D babel-jest @babel/core @babel/preset-env
```

**npm i -D babel-plugin-transform-es2015-modules-commonjs**

```
PS C:\Users\1701805\Documents\_D2WM\_02-MOD-FRONT-END\01-SEQ-REALISER-UNE-INTERFACE\_SEA-03-DEVELOPPER\_JEST\HELLO_JEST> npm i -D babel-plugin-transform-es2015-modules-commonjs
```

Maintenant, si vous ouvrez le fichier **package.json**

```
"devDependencies": {
  "@babel/core": "^7.16.12",
  "@babel/preset-env": "^7.16.11",
  "@types/jest": "^27.4.0",
  "babel-jest": "^27.4.6",
  "babel-plugin-transform-es2015-modules-commonjs": "^6.26.2",
  "jest": "^27.4.7"
}
```

Vous voyez les plugins Babel installés dans la partie devDependencies

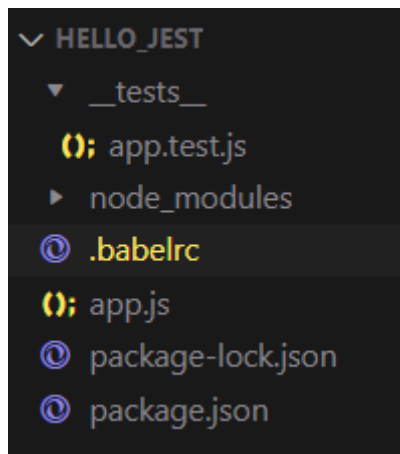
Ensuite il vous faut ajouter un fichier de configuration Babel : ".babelrc" (attention au point devant le nom de fichier).

## Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021



Ce fichier de configuration doit contenir ce qui suit :

```
© .babelrc > {} env > {} test
1  {
2    "env": {
3      "test": {
4        "plugins": ["transform-es2015-modules-commonjs"]
5      }
6    }
7  }
```

`{ "env": { "test": { "plugins": ["transform-es2015-modules-commonjs"] } } }`

Ensuite dans **app.js** :

```
export default{
  containForbiddenWords,
  changeForbiddenWords
}

/*module.exports = {
  sentence,
  isPalindrome,
  lettersNumber,
  containForbiddenWords,
  changeForbiddenWords
}*/
```

Puis dans **app.test.js** :

#### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021

```
_tests_ > (); app.test.js > ...  
1 //const app = require('../app');  
2 import app from '../app.js';  
3  
4
```

Vous pouvez lancer un test pour vérifier que tout fonctionne

#### Développeur web et web mobile

**CCP** : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

**MESM** : Développer une interface utilisateur web dynamique en effectuant une veille technologique y compris en anglais

Réalisation Stéphane Pontonnier – 2021