Les tests en JavaScript

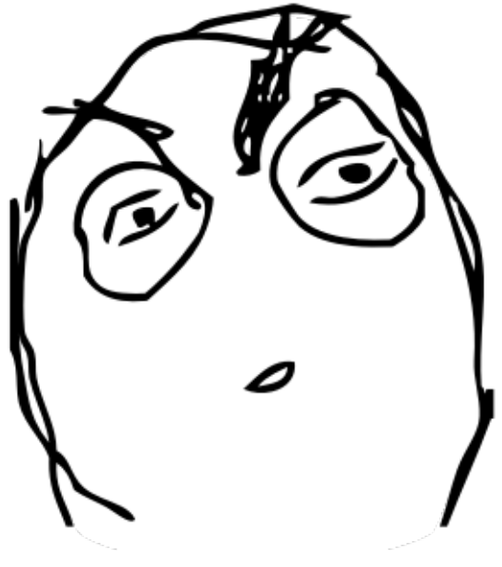
Alvin Berthelot

Version 1.0.0

La licence, ses explications ainsi que les moyens de contribution et réappropriation sont détaillés à la fin.

Les tests …

pourquoi déjà ?



Définition d'un test

Un test correspond à la **validation d'un comportement attendu**.

Pour être efficace et efficient :

- Un test doit être **RAPIDE** à s'exécuter.
- Un test doit être **FACILE** à comprendre.
- Un test doit être isolé, bref **PRÉDICTIBLE**.

Bénéfices de l'éxécution des tests

S'assurer de la non régression d'une application si celle-ci s'enrichit de fonctionnalités ou est refactorée.

Doit permettre de tester de **multiples environnements d'exécution**.

Doit permettre de **vérifier la couverture du code et de sa lisibilité**.

Bénéfices de l'écriture des tests

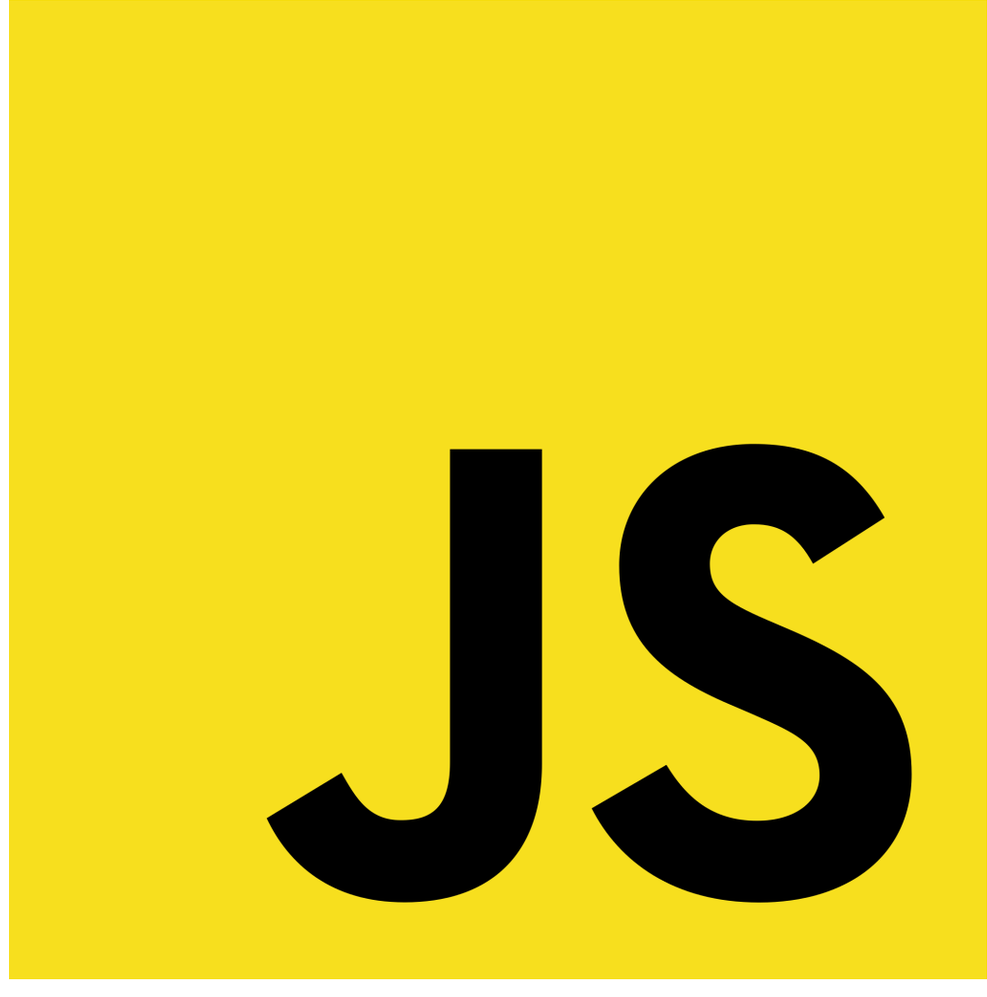
Peut servir de **documentation sur les objectifs attendus** de l'application.

Peut **guider dans le développement d'un algorithme**.

S'apercevoir qu'il est difficile d'écrire un test, c'est déjà **s'apercevoir d'un "Code smell"**.

5 fruits et légumes par jour

Et JavaScript ?



S'essayer aux tests … en JavaScript

S'essayer aux tests, c'est déjà comprendre le jargon associé :

- Types de test : **test unitaire**, **test d'intégration**, **test E2E** ("end-to-end").
- Analyse de la **qualité du code** et la **couverture du code** testé.
- **Savoir faire semblant** : "Spies", "Stubs", "Mocks".

Si en plus on est dans l'univers de JavaScript :

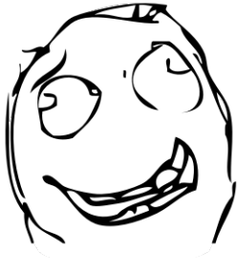
- Ce **langage si permissif**.
- Vive les **environnements d'exécution** !
- La **multitude des micro-librairies**.

Restons pragmatique

On fait quoi avec JavaScript aujourd'hui ?

Il y a de forte de chance de vous retrouver dans un de ces 3 cas de figure, votre travail c'est de :

1. **Développer une application front-end**.
2. **Développer une application back-end**.
3. **Développer une librairie**.



Tester une librairie, cas simple

Une librairie au fond c'est une multitude de fonctions. **Une (bonne) fonction en soit c'est très simple à tester**.

Il faut juste **lancer des tests** et **vérifier que les résultats obtenus sont les bons**.

Bref il vous faut un "runner" et une librairie d'assertion. Les 2 options les plus répandues :

- **Jasmine** fait les 2 à la fois.
- **Mocha** et **Chai** est un super combo.

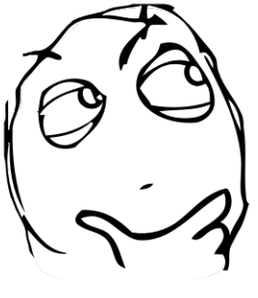
Tester une librairie, cas plus complexe

Ok mais ce fameux **problème des différents environnements d'exécution** ?

Côté back-end, c'est du Node.js pas de recette miracle, il vous faudra plusieurs versions et les lancer sur chaque version.

Côté front-end, c'est le "runner" **Karma** qui va s'occuper de lancer Chrome, Firefox, etc. et de lancer vos tests … toujours pas de recette miracle, il vous faudra plusieurs navigateurs et plusieurs versions.

RAPIDE ? FACILE ? PRÉDICTIBLE ?



Tester une application back-end, cas simple

Un back-end au fond ce sont des **services auxquels on peut accéder à distance** (en général via HTTP) et qui ont **des effets de bord** (en général une base de données).

Bref il vous faut une librairie de requêtage HTTP en plus. Les 2 options les plus répandues :

- SuperAgent.
- supertest.

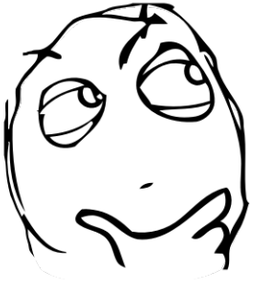
Tester une application back-end, cas plus complexe

Ok mais ces fameux **effets de bord** ?

Si lancer des tests dans un contexte particulier ou ayant des effets de bord n'est pas quelque chose de douloureux, alors autant faire avec, c'est plus réaliste.

Sinon et bien il faut utiliser Sinon.js qui va savoir faire semblant et simuler un contexte et les effets de bord associés afin de mieux isoler votre test.

RAPIDE ? FACILE ? PRÉDICTIBLE ?



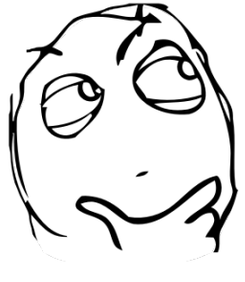
Tester une application front-end

Soit vous voulez tester des "composants" isolés/isolables : composant graphique, service, modules, etc. et du coup à peu de choses près vous allez retomber sur les mêmes démarches que pour tester une librairie ou un back-end.

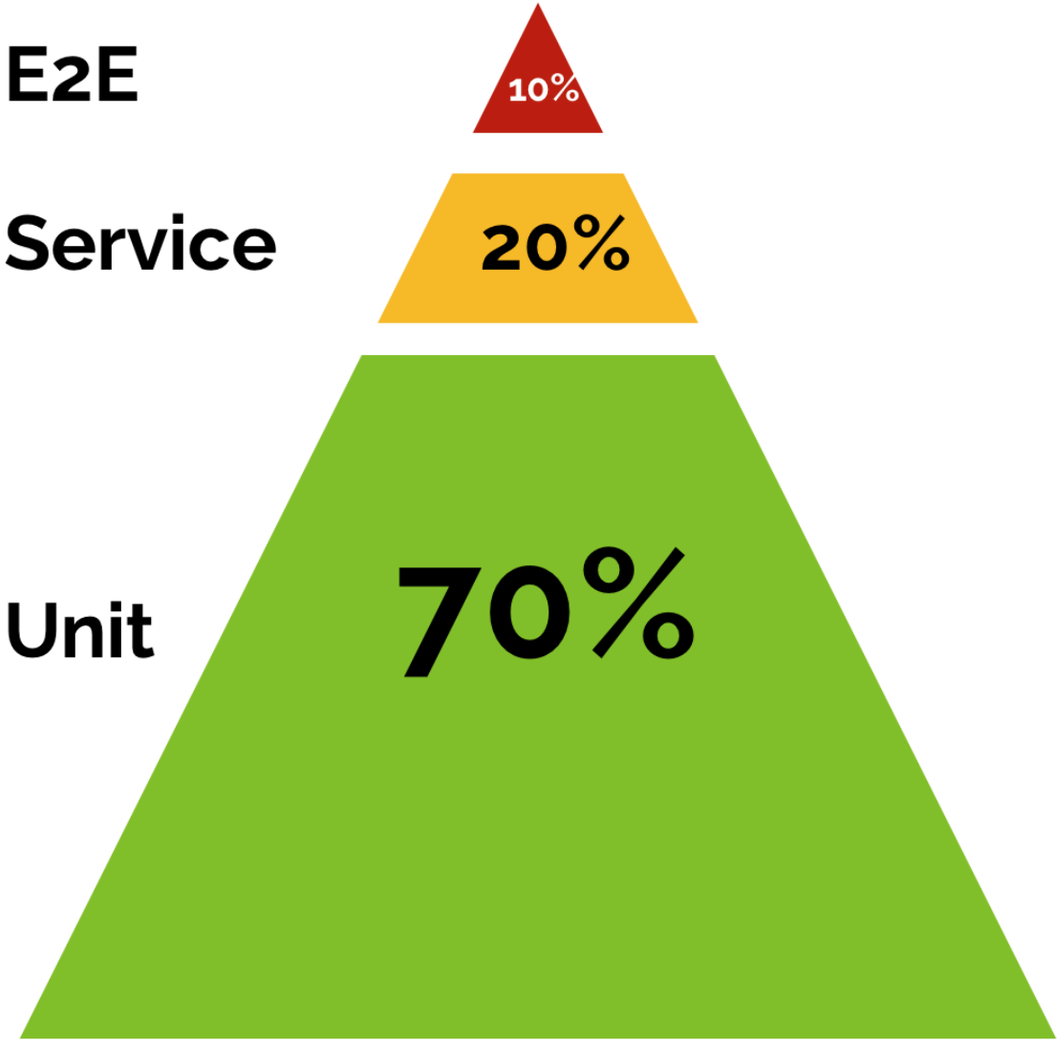
Soit vous voulez tester les comportements d'un point de vue purement IHM : événements, modification du contenu HTML. Là vous allez devoir utiliser Selenium, euh non pardon je voulais dire Protractor.

La question du E2E va arriver, dois-je tester mon application de bout en bout ?

RAPIDE ? FACILE ? PRÉDICTIBLE ?



La pyramide de tests de Mike Cohn



Qualité et couverture du code

C'est déjà un bon panorama : Mocha, Chai, Karma, SuperAgent, Sinon, Protractor.

Sauf que je n'ai pas trop abordé la qualité et la couverture du code …

Pour la qualité, **un linter**, genre ESLint.

Pour la couverture du code, **un reporter**, genre Istanbul.

Là pour le coup **pas besoin de réflechir c'est RAPIDE, FACILE, PRÉDICTIBLE**.

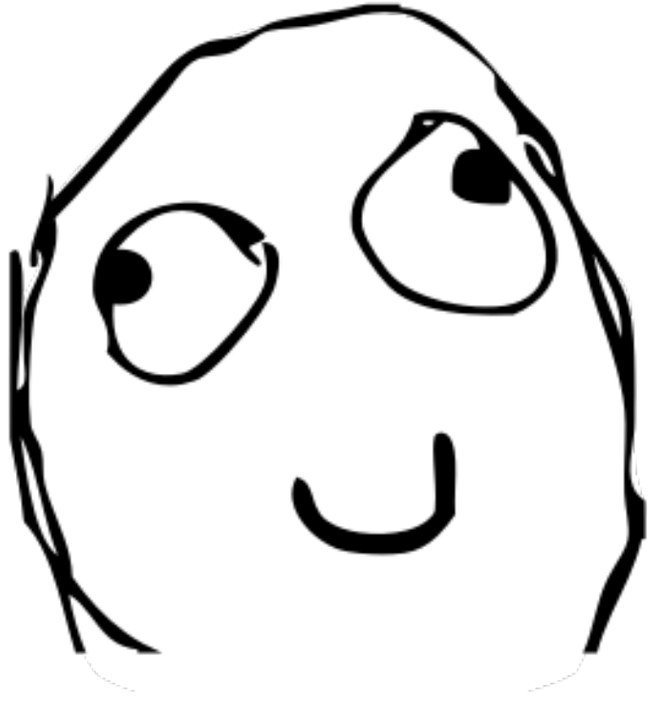
Bilan

Ce n'est qu'un rapide panorama des outils et pratiques, il n'y a pas de chemin tout tracé, **c'est à vous de réflechir sur le ROI** (tests manuels Vs tests automatisés).

Garder toujours en tête ce tryptique : **RAPIDE ? FACILE ? PRÉDICTIBLE ?**, car c'est votre fil conducteur pour vos tests mais aussi (voir surtout) pour votre code.

Bien évidemment il faudra mettre ça dans une **intégration continue** : Jenkins, Travis.

MERCI !



Licence

Pour toutes questions, réclamations ou remarques, merci d'envoyer un message à alvin.berthelot@webyousoon.com.

Explications licence CC BY-SA 3.0

Cette licence permet aux autres de remixer, arranger, et adapter votre œuvre, même à des fins commerciales, tant qu'on vous accorde le mérite en citant votre nom et qu'on diffuse les nouvelles créations selon des conditions identiques.

Cette licence est souvent comparée aux licences de logiciels libres, "open source" ou "copyleft".

Toutes les nouvelles œuvres basées sur les vôtres auront la même licence, et toute œuvre dérivée pourra être utilisée même à des fins commerciales.

C'est la licence utilisée par Wikipédia ; elle est recommandée pour des œuvres qui pourraient bénéficier de l'incorporation de contenu depuis Wikipédia et d'autres projets sous licence similaire.

Contribution et réappropriation

Ce fichier PDF est généré avec Asciidoctor à partir d'un dépôt Git se trouvant sous GitHub.

https://github.com/alvinberthelot/slides-js-testing

Cela signifie que vous n'avez pas besoin de vous battre avec un fichier binaire (le PDF) pour **contribuer**, **vous réapproprier le contenu** ou **modifier le thème** de présentation.

Contribution

Vous voulez **contribuer au contenu** car :

- Il y a une erreur (ça arrive à tout le monde), de typographie, de compréhension, ou tout autre chose.

- Vous souhaitez apporter une précision.

Il vous suffit de contribuer au projet via Git par le moyen d'une "pull request" sur le dépôt Git.

Réappropriation

N'oubliez pas les conditions de la licence.

Vous voulez vous **réapproprier le contenu** car :

- Vous souhaitez donner un style différent.
- Vous souhaitez enlever/ajouter/modifier des sections dans votre contexte.

Il vous suffit de "forker" le dépôt Git et d'y apporter vos propres modifications, puis de générer par vous même le nouveau PDF.