ARTRAGIS

Zeste de savoir

[Humeur] Les tests sur jest : les snapshot

13 décembre 2022

Table des matières

Introduction

Il y a quelques temps maintenant, lors du développement initial de zMarkdown, @cepus a mis en place une suite de tests assez complète.

Au vue de l'architecture de zMarkdown, les tests sont placés à deux endroits:

— dans chaque plugin: on teste le plugin de manière générale, avec des données assez peu pertinentes mais qui représentent les cas unitaires qu'on veut. C'est important car les plugins peuvent être utilisés par d'autres projets opensource
— dans zmd: on met une suite complète, parfois directement citée des tutos de zds, avec donc des corpus assez massif. C'est important car ça nous assure que zmd marche avec ce que font les membres du sites.

Le problème est donc ici double: on a **beaucoup** de tests, tous basés sur du texte et à chaque texte l'input et l'ouput peuvent potentiellement faire des dizaines de lignes de markdown, de html ou de CSS.

C'est là qu'entrent en jeu les snapshot.

1. Un snapshot, comment ça se fait ?

Avec jest, rien de plus simple:

```
test('footnotes', () => {
  const p = renderString(dedent`
    # mytitle A[^footnoteRef]

    [^footnoteRef]: reference in title

    # mytitle B^[footnoterawhead inner]

    # myti*tle C^[foo inner]*
```

```
10
11      a paragraph^[footnoteRawPar inner]
12    `)
13    return expect(p).resolves.toMatchSnapshot()
14  })
```

Listing 1 – exemple de test jest sur zmarkdown: on transforme le markdown puis on fait `expect(actual).resolves.toMatchSnapshot()`

Lorsqu'on exécute `jest` pour la toute première fois le test est au vert.
Si on modifie le code, il vous dira que ça ne matche plus et qu'il faut soit mettre à jour le snapshot, soit modifier le code.

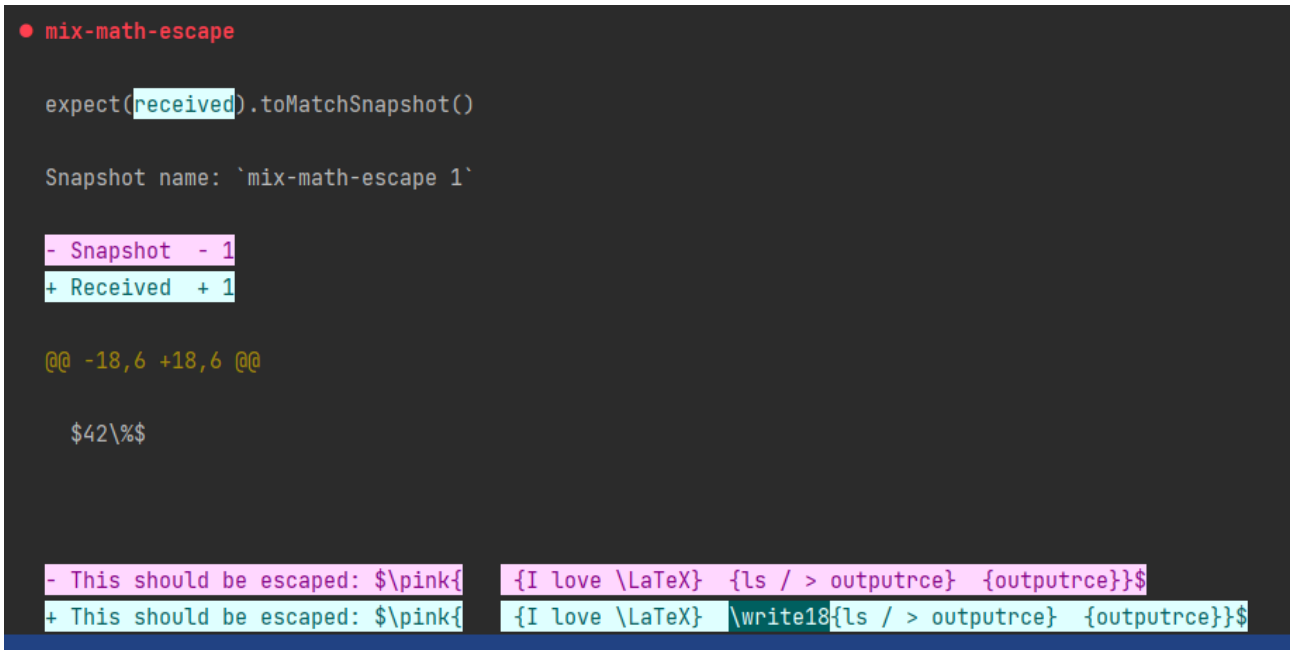

FIGURE 1.1. – Le résultat d'un fail de snapshot dans jest

Tout cela parait incroyablement smooth a première vue: lorsque le test évolue, il suffit de valider que les différences sont bonnes. Si tel est le cas on lance `jest -u` sinon on modifie le code.
En plus avec toute cette mise en forme on sait ce qui a changé. Bref, lorsque tout cela a été mis en place, j'avais des étoiles dans les yeux. Juste une petite appréhension: lorsque je débug j'ai l'habitude d'écrire le test **avant**, de vérifier qu'il échoue puis de corriger.
Dans le cas présent, le test n'échouera pas au début, ou alors il faut que j'aille modifier à la main le snapshot. Qui ressemble à ça:

```
1  exports[`footnotes 1`] = `
2  "\\\\levelOneTitle{mytitle A\\\\textsuperscript{\\\\protect\\\\footnotemarl
3
4
5  \\\\footnotetext[1]{reference in title}
6
```

```
7  ⌋
       \\\\levelOneTitle{mytitle B\\\\textsuperscript{\\\\protect\\\\footnotemark
8
9
10 ⌋
       \\\\levelOneTitle{myti\\\\textit{tle C\\\\textsuperscript{\\\\protect\\\\fo
11
12
13 ⌋
       a paragraph\\\\textsuperscript{\\\\footnotemark[4]}\\\\footnotetext[4]{foo
14 `;
```

Listing 2 – oui, débuger un snapshot latex, c'est compliqué

2. Grrr

Ma petite apréhension va vite devenir un cauchemard.
Si dans l'exemple que je vous ai donné, le travail semble accessible, c'est principalement car le corpus est court et que chaque ligne est bien séparée.
Maintenant imaginons un corpus propre de ce qu'on attend sur zds, avec par exemple un tableau complexe ou encore une vraie formule de math ?
Le snapshot devient vite illisible et comme la première fois qu'on a ajouté des tests il est passé au vert… Pas facile de voir qu'il y avait une faute.
Prenez par exemple, pour éviter une faille de sécurité, nous devons faire en sorte que la commande suivante soit proprement échappée ou du moins que les parties dangereuses soient retirées: `This should be escaped: $\pink{\text{I love \LaTeX} \immediate\write18{ls / > outputrce} input{outputrce}}$.`
Cette phrase est importante car elle permet de tester un cas de formule inline, au sein d'une vraie phrase, où il y a plusiers commandes imbriquées et où la faille de sécurité est basée sur une double commande latex.
Lorsque je développais, j'avais oublié de prendre en compte ce cas de "double commande" donc c'est très bien qu'un teste existe et le montre. Cependant, lorsque j'ai exécuté mon test, je me suis retrouvé face à un problème: au départ, les commandes `pink` et `latex` étaient aussi retirées.
Résultat, alors même que j'avais bel et bien corrigé le bug, le test ne passait pas. J'ai un peu tourné en rond pour comprendre pourquoi je faisais mal les retraits, avant de comprendre que c'était parceque le snapshot était mauvais. Il a donc fallu rembobiner mon code, corriger le snapshot, puis valider que ça fonctionnait.
Le screenshot que j'ai placé en début de billet montre justement l'étape après rembobinage et correction du snapshot.
J'ai donc perdu beaucoup de temps.

Conclusion

Bref, j'aime pas les snapshots, ou du moins j'ai perdu mon enthousiasme à leur égard.
Avez-vous, vous aussi un petit retour d'expérience à leur propos ?