Manuel InsideTest

InsideTest permet de faire des tests à la fin des modules javascript, comme avec ruby par exemple.

Le fonctionnement est cependant assez différent des autres modules de test, il est spécialement pensé pour les tests répétitifs, c'est-à-dire qui testent le résultat de nombreuses valeurs. L'idée est alors que créer un test, puis de passer ces valeurs (appelées `sujets`) par ce test.
Par exemple :

```
// Un test qui doit vérifier que les valeurs sont bien égales à 4
var test = new InsideTest({
  error: '%{devrait} être égal à 4.'
  eval: function(sujet){
    return sujet == 4
  }
})
// On fait les tests
test.with(2) // => une erreur en console
test.with(3) // => idem
test.with(4) // succès (invisible par défaut)
test.with(2 + 2) // => succès

test.withNegate(4) // => Erreur "4 ne devrait pas être égal à 4."
test.withNegate(3) // => succès
```

Définition des messages

Comme on peut le voir ci-dessus, on définit simplement les messages d'erreur à l'aide d'un texte simple qui doit pouvoir marcher (mais pas forcément) avec les tests négatifs. On utilise pour ça la valeur template `%{devrait}` ou `%{doit}` qui suivant le résultat se transformera en "devrait"/"ne devrait pas" ou "doit"/"ne doit pas".

Les tests ne produisent aucun retour en cas de succès, ils ne sont là que pour déceler les problèmes.

Activation des tests

Pour activer les tests, il faut mettre une constante `INSIDE_TESTS` à la valeur `true` et ce au plus haut du programme, dans une balise `<script>` en dur. Car les tests sont joués au chargement des modules.

Il est d'ailleurs préférable que les modules javascript auto-testés soient chargés avec un `defer` :

```
<script type="text/javascript" src="path/to_script" defer></script>
```

Les méthodes de test

.with(sujet[, expected])

Quand on veut seulement tester un sujet.

Si `expected` est défini, la méthode se comporte comme `.withExpected`

Bien prendre en compte les problèmes d'égalité.

.withNegate(sujet[, expected])

Inverse de la précédente.

.withExpected(sujet, expected)

Fournit le `sujet` à la méthode de test et espère le résultat `expected`. Bien prendre en compte les problèmes d'égalité.

Exemple :

```
var test = new InsideTest({
    error: '%{devrait} correspondre.'
  , eval: function(sujet){
      return `Bonjour, ${sujet} !`
    }
})

test.withExpected('John', 'Bonjour, John !') // => succès
test.withExpected('Renée', 'Bonjour, Renée !') // => succès
test.withExpected('Al', 'Au revoir, Al')
// => échec. En console :
// «« Al »» devrait correspondre.
// Attendu : "Au revoir, Al"
// Obtenu  : "Bonjour, Al !"
```

.withExpectedNegate(sujet, expected)

Inverse de la précédente.

.equal(sujet, expected)

Teste l'égalité entre `sujet` et `expected`.

Cette méthode prend en compte les problèmes d'égalité.

Annexe

Problèmes d'égalité

Avec Javascript, les égalités ne sont pas pratiques... Par exemple, `[1,2,3]` ne sera pas égal à `[1,2,3` ...

Quand on veut tester une telle égalité, soit on utilise `JSON.stringify` (ou `JString(…)`) à l'intérieur de la fonction :

```
var test = new InsideTest({
  eval: function(sujet, expected){
    // ...
    return JString(sujet) == JString(expected)
  }
})
```

... Soit on utilise la méthode de test `equal`.