JavaScript をテストするときに考える10のことがら+1

10+1 THINGS YOU SHOULD KNOW ABOUT JAVASCRIPT TESTING

Takuto Wada (@t_wada on Twitter)

Nov 20, 2010 @ Firefox Developers Conference

Who am I ?

- TDD guy in Japan.
- @t_wada on Twitter
- twada on github

大事なことを最初に

=> 宣伝

First things first.

=> Ad.

日本人寄稿者
(邦訳オンリー)
@omo2009
@m_seki
@hyoshiok
@miyagawa
@hotchpotch
@dankogai
@yukihiro_matz
@t_wada

97 Things Every Programmer Should Know

日本人寄稿者
(邦訳オンリー)
@omo2009
@m_seki
@hyoshiok
@miyagawa
potch
nkogai
matz
t_wada

12月中旬発売

97 Things Every Programmer Should Know

O'REILLY
オライリー・ジャパン

Kevlin Henney 編
和田 卓人 監修
夏目 大 訳

よろしくおねがいします

YOROSHIKU ONEGAI SHIMASU

1.「全部をテストする」ことはできない

YOU CAN'T TEST "EVERYTHING".

「全部をテストする」ことはできない

- 使い心地、綺麗な見た目のテストは人間にしか出来ない
- Usability, look and feel can be tested only by manually.

「全部をテストする」ことはできない

- 「人間でなくとも出来ることをテストにサポートさせる」と考える
- Other aspects can be tested automatically, so let computers do them.

2. 目視を可能な限り減らす

REDUCE VISUAL INSPECTION AS MUCH AS POSSIBLE.

目視を可能な限り減らす

- 目視必須の所と目視不要の所を分ける。
- Separate code that must be inspected visually from other code.

目視を可能な限り減らす

- 目視不要の部分が増えるほど、人間の手間が減る
- The more you automate the non-visual testing, the more benefit you gain.

3. 同期処理と非同期処理を分ける

DIVIDE SYNCHRONOUS PART FROM ASYNCHRONOUS PART.

同期処理と
非同期処理を分ける

- 非同期処理はテストが難しく、テスト実行時間もかかりがち
- Asynchronous tests tend to be complex and time-consuming.

- コードの非同期処理部分(イベントハンドラ)を最小にして、同期的にテストできる部分を増やす
- Maximize amount of code that can be tested synchronously, by minimizing asynchronous part of code. (e.g., event handlers)

4. 内部構造を隠蔽しすぎない

AVOID TOO MUCH INFORMATION HIDING.

内部構造を
隠蔽しすぎない

- 例えば、テストですり替えたいハンドラに匿名関数を使わない
- For example, avoid using anonymous functions for event handlers.

- 内部の接合点をテストから見えるように、差し替えられるように設計する
- Pay attention to “Seams” and “Interception Points”. Provide enabling points for Mocks/Stubs. (see WEwLC)

5. カスタムイベントで粗結合化する

Decouple components by using custom-events.

カスタムイベントで粗結合化する

- イベント発火元と受取り側の結合度を減らす
- Decouple event handlers from event sources, by using custom events.

カスタムイベントで粗結合化する

- デザイン変更、DOM構造の変更からロジックを隔離する
- Make code independent from Design/DOM structure change.

6. 先人の設計に学ぶ

STANDING ON THE
SHOULDERS OF GIANTS.

先人の設計に学ぶ

- 書籍を紐解き、先人の知恵に学ぶ
- Read good books. Learn the wisdom and culture from these books.

先人の設計に学ぶ

- GoF のパターンは UI 開発に使えるものが多い
- GoF patterns are good for rich user interface code design.

7. 迷ったら、シンプルな仕組みを好む

IF IN DOUBT,
PREFER SIMPLER SOLUTION.

迷ったら、シンプルな仕組みを好む

- 複数の解法やライブラリなどで迷ったら、シンプルな方を選ぶ
- If there are two or more ways of solving problems, prefer simpler one.

迷ったら、シンプルな仕組みを好む

- jQuery, そして QUnit もそう
- Simpler solution may survive. jQuery and QUnit have quality and beauty of simplicity.

8. ソフトウェアの梃子(てこ)の効果を生かす

USE SOFTWARE LEVERAGE TO YOUR ADVANTAGE.

ソフトウェアの梃子(てこ)の力を生かす

- なるべく車輪の再発明はしない(車輪の再開発/実装はしてもいい)
- Don’t reinvent the wheels. (But re-implement them sometimes)

- UNIX 文化を尊重し、CUI を使う。プレーンテキスト、スクリプト、パイプ/フィルタを使いこなす
- Prefer CUI. Respect UNIX culture. Use the power of plain text format, scripting, pipes and filters.

9. ブラウザを使わなくてもテストできるようにする

SEEK FOR THE WAY OF TESTING OUTSIDE THE BROWSER.

ブラウザを使わなくてもテストできるようにする

- env-js や xmlw3cdom.js は、ブラウザや DOM のふりをする
- Try to use fake-browser-like libraries. See env-js and xmlw3cdom.js

QUnit-TAP というものを作りました

- QUnit-TAP (my product) produces TAP output from QUnit test code.
- http://github.com/twada/qunit-tap

10. 徹底的に自動化する

AUTOMATE MERCILESSLY.

徹底的に自動化する

- 継続的インテグレーション：自動テストは定期的に実行させる
- Continuous Integration : Run automated tests periodically

徹底的に自動化する

- hudson かわいいよ hudson
- hudson! hudson!

11. 未だフロンティアである

STILL IN FRONTIER.

未だフロンティアである

- JavaScript のテストの決定解はいまだ無い
- Still no definitive solutions for JavaScript testing.

未だフロンティアである

- JavaScript テストの未来を作るのは、あなたかもしれない
- YOU may be the one. Create the future of JavaScript testing.

ご清聴

ありがとう

ございました

THANK YOU VERY MUCH.

延長戦

EXTRA TIME.

デモ：DEMO

- Spidermonkey + QUnit + QUnit-TAP + Growl

- http://github.com/twada/qunit-tap

デモ：DEMO

- Spidermonkey + QUnit + QUnit-TAP + xmlw3cdom.js

- http://github.com/twada/qunit-tap