Documenting behaviour with tests

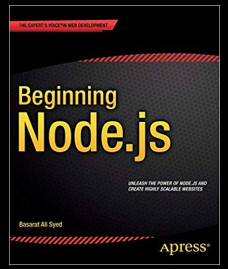
—

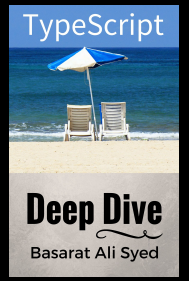
 @basarat

About me

Microsoft® MVP Most Valuable Professional

AUSTRALIA POST





/basaratali

/basarat

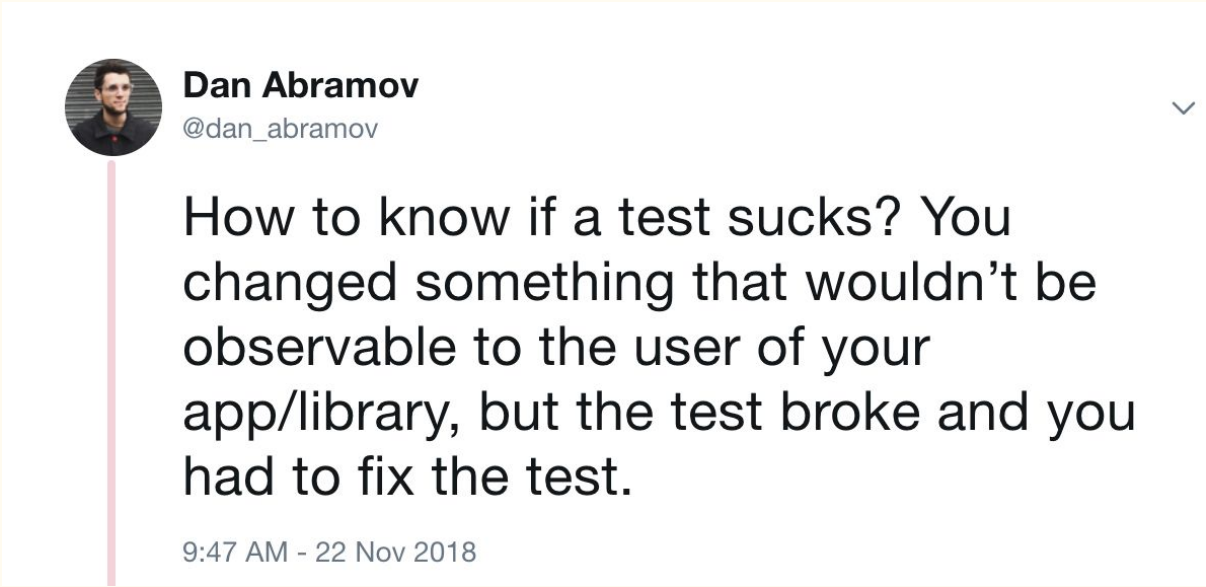
@basarat

basarat.com

Accompanying Code

All the code from the demos:
https://github.com/basarat/2019-holyjs

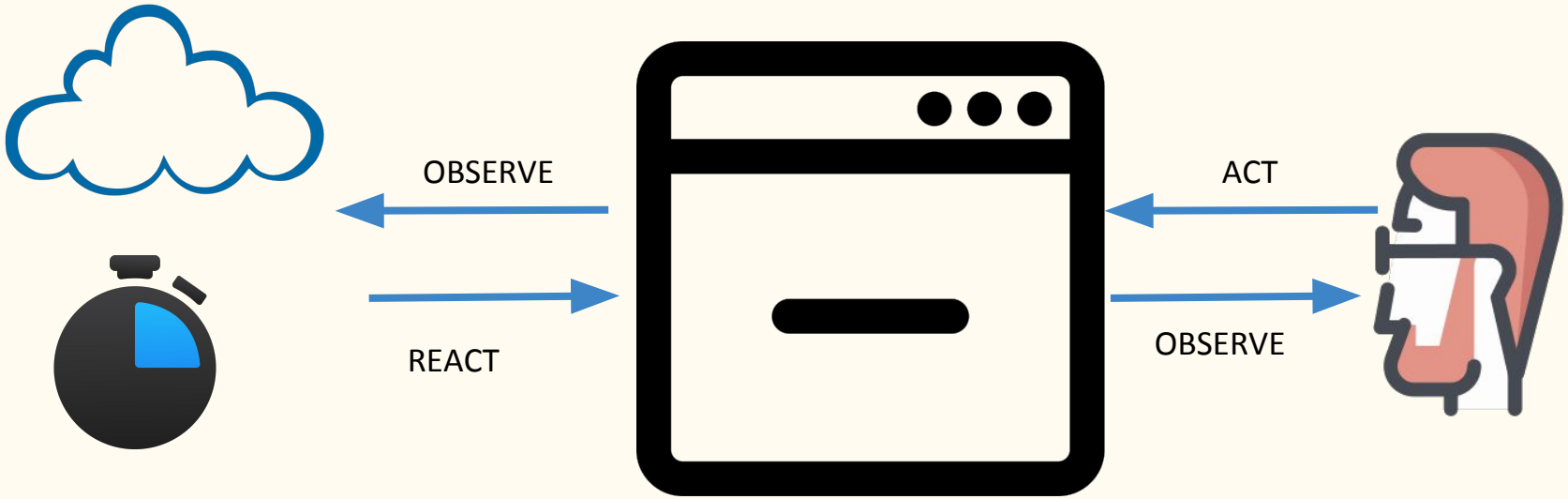
Test behaviour
Not implementation


Dan Abramov
@dan_abramov
How to know if a test sucks? You changed something that wouldn't be observable to the user of your app/library, but the test broke and you had to fix the test.
9:47 AM - 22 Nov 2018

reference

Tests serve a purpose

Document how should the UI behave


OBSERVE
REACT
ACT
OBSERVE


cypress.io

- beforeEach / it abuse
- detectChanges
- delay(random)
- debugging failures

- Command-execution separation
- Retry command till it succeeds
- DOM hooks on getters
- DOM snapshots / Video

Cypress Concepts Demo


TodoMVC

Functionality

No todos

When there are no todos, `#main` and `#footer` should be hidden.

New todo

New todos are entered in the input at the top of the app. The input element should be focused when the page is loaded, preferably by using the `autofocus` input attribute. Pressing Enter creates the todo, appends it to the todo list, and clears the input. Make sure to `.trim()` the input and then check that it's not empty before creating a new todo.

Mark all as complete

This checkbox toggles all the todos to the same state as itself. Make sure to clear the checked state after the "Clear completed" button is clicked. The "Mark all as complete" checkbox should also be updated when single todo items are checked/unchecked. Eg. When all the todos are checked it should also get checked.

Item

A todo item has three possible interactions:

1. Clicking the checkbox marks the todo as complete by updating its `completed` value and toggling the class `completed` on its parent `<li>`
2. Double-clicking the `<label>` activates editing mode, by toggling the `.editing` class on its `<li>`
3. Hovering over the todo shows the remove button (`.destroy`)

Editing

When editing mode is activated it will hide the other controls and bring forward an input that contains the todo title, which should be focused (`.focus()`). The edit should be saved on both blur and enter, and the `editing` class should be removed. Make sure to `.trim()` the input and then check that it's not empty. If it's empty the todo should instead be destroyed. If escape is pressed during the edit, the edit state should be left and any changes be discarded.

Counter

Displays the number of active todos in a pluralized form. Make sure the number is wrapped by a `<strong>` tag. Also make sure to pluralize the `item` word correctly: `0 items` , `1 item` , `2 items` . Example: **2** items left

Clear completed button

Removes completed todos when clicked. Should be hidden when there are no completed todos.

Persistence

Your app should dynamically persist the todos to localStorage. If the framework has capabilities for persisting data (e.g. Backbone.sync), use that. Otherwise, use vanilla localStorage. If possible, use the keys `id` , `title` , `completed` for each item. Make sure to use this format for the localStorage name: `todos-[framework]` . Editing mode should not be persisted.

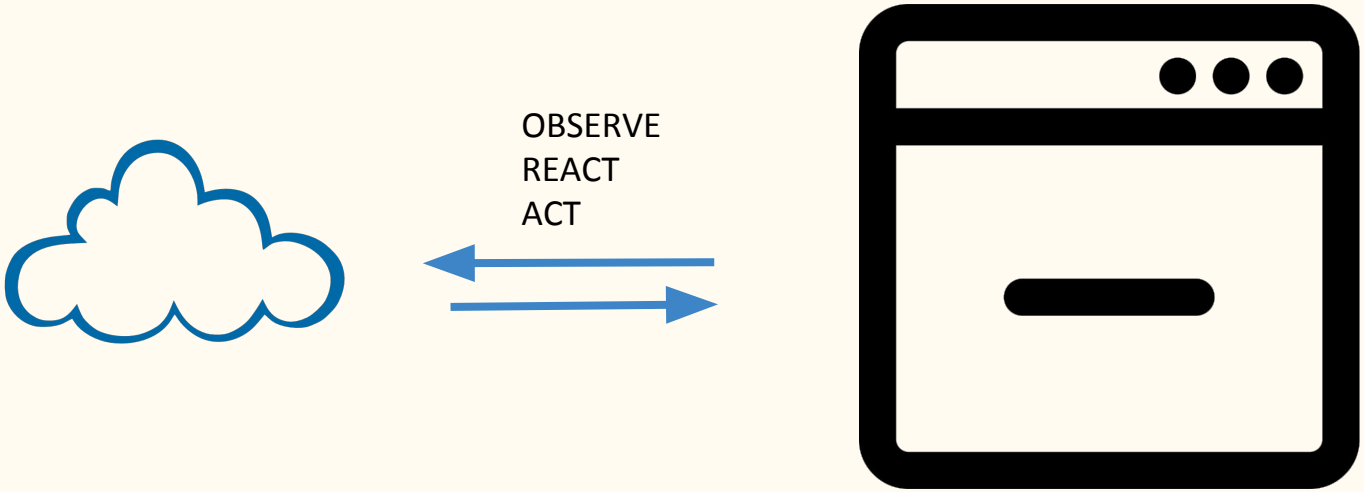
Routing

Routing is required for all implementations. If supported by the framework, use its built-in capabilities. Otherwise, use the Flatiron Director routing library located in the `/assets` folder. The following routes should be implemented: `#/` (all - default), `#/active` and `#/completed` (`#!/` is also allowed). When the route changes, the todo list should be filtered on a model level and the `selected` class on the filter links should be toggled. When an item is updated while in a filtered state, it should be updated accordingly. E.g. if the filter is `Active` and the item is checked, it should be hidden. Make sure the active filter is persisted on reload.

https://github.com/tastejs/todomvc/blob/master/app-spec.md

Demo

Backend Dependency


OBSERVE
REACT
ACT

Backend Demo

Page Objects

Demo

Behaviour Spec
Vs.
Implementation Spec

Item

A todo item has three possible interactions:

1. Clicking the checkbox marks the todo as complete by updating its `completed` value and toggling the class `completed` on its parent `<li>`
2. Double-clicking the `<label>` activates editing mode, by toggling the `.editing` class on its `<li>`
3. Hovering over the todo shows the remove button (`.destroy`)

Editing

When editing mode is activated it will hide the other controls and bring forward an input that contains the todo title, which should be focused (`.focus()`). The edit should be saved on both blur and enter, and the `editing` class should be removed. Make sure to `.trim()` the input and then check that it's not empty. If it's empty the todo should instead be destroyed. If escape is pressed during the edit, the edit state should be left and any changes be discarded.

Item

- Starts of unchecked
- Clicking the checkbox toggles the todo active/complete
- Clicking the remove button should remove it item

Edit item

- Double-clicking the todo label activates editing mode
- The edit mode should exit on enter, blur and escape
- Enter results in a commit
- Blur results in a commit
- The *commit* is done after trim
- If the trim results in an empty value, the commit should destroy the item
- Escape does not result in a commit

Let’s write some docs*

* code

Thank You

https://basarat.com
https://youtube.com/BasaratAli

@basarat