2020-08-11 UI Testing Team Meeting Notes

Date
11 Aug 2020

Attendees
- Anton Emelianov
- Zak_Burke
- Mike Taylor
- John Coburn
- Charles Lowell
- Taras Mankovski
- Viktor Soroka

Goals
- Review Cypress and BigTest 1.0 (new version)

Recording
or
https://drive.google.com/file/d/196Qk5GJPMCK0UtBeH0ohW-14kHP8LE7n/view?usp=sharing

Discussion items

<table>
<thead>
<tr>
<th>Time</th>
<th>Item</th>
<th>Who</th>
<th>Notes</th>
</tr>
</thead>
</table>
| 30 min | Cypress Demo        | Mike Taylor |  - ui-courses uses cypress and yakbak; are orthogonal  
  - yakbak: from flickr  
    - annoying: expects tests to KNOW they're running with yakbak; ditto polly  
    - yakbak-proxy: runs tapes via a proxy to work around that annoying bit  
  - counts put/post/delete requests in order  
  - cypress: open  
    - in Jenkins, runs in headless mode BUT!! has screen caps for failed tests, and video  
    - mocha is the assertion library  
    - cy.* is analogous to nightmare.click, nightmare.wait, etc  
    - cypress is asserting lots of things behind the scenes  
    - implicit wait, a la convergences, is 4 seconds  
    - "selector playground" can suggest use  
    - helpers: Cypress.Command.add to add helper function to cy"  
    - WRT documentation: "by some distance, the best ever"; active Gitter channel  
    - see also: cypress 'testing-library, a unit-testing companion to
<table>
<thead>
<tr>
<th>John Coburn</th>
<th>Charles Lowell</th>
</tr>
</thead>
</table>

- **BigTest 1** uses Karma
  - BigTest 2 has replaced Karma with their own runner
- installed deps, config file, tweak things,
- emphasis on succinct syntax:
- good selectors, easy to e.g. test
- versatility wins: real browsers, AND JSDom/Node
- mocking of backend is an orthogonal concern
- functions similar to cypress: backend is a blackbox
- FS Runner, analogous to Cypress’s
- agents for Chrome, Firefox, Safari
  - can point at any URL you want, i.e. can function for e2e tests
- any browser that can load a URL can connect to BigTest as an agent
- tests separated into tests/assertions
  - tests move state of app forward
  - assertions are pure, cannot change state
  - lessons both in usage and expression
- borrows from Capybara
  - interactor sits between DOM and user
  - no more decorators: very functional composition of interactors
  - gives good error messaging when things fail
- interactor: defaultSelector is the "human readable" match
  - BUT can use any kind of match, e.g. name, id, aria-label etc
- mutating steps/pure assertions is very powerful, leads to easy auto optimizations w/in suite
- mutating actions are always evaluated so never have stale refs
- TypeScript! So you get intellisense when writing interactors
- BT runs a proxy in between running service and test runner
- BigTest 1 was really just a collection of tools with siloed data and therefore poor interactions among them
- BigTest 2: reason about entire test stack as a single state; NO SILOs
  - very few deps ... but is all homegrown :)
  - partnered with Jonas Nicklas (capybara creator; https://twitter.com/jonicklas?lang=en)
- pillars of architecture
  - data: all test result/assertion data available as graphql
  - structured concurrency: cleanup correctly
  - native agents: comm with browser via WebSockets
  - i.e. tests are represented as data in GraphQL
  - representing assertion as a piece of data is very powerful
    - can get stack trace, detailed information
    - can write agent in
  - this is all OSS
- tests as data, not as scripts
  - note: structure is an exported structure
  - it's up to the agent to interpret the test
  - so can layer on new syntax, b/c just maps onto a data structure
    - e.g. cuke syntax is on the way because can just map Gherkin onto BT structure

---

**Action items**

- [ ]