(((IN/Clojure)))

Nov 26, 2016

Designing an "Object-Functional" System with Clojure

@AdityaAthalye

helpshift

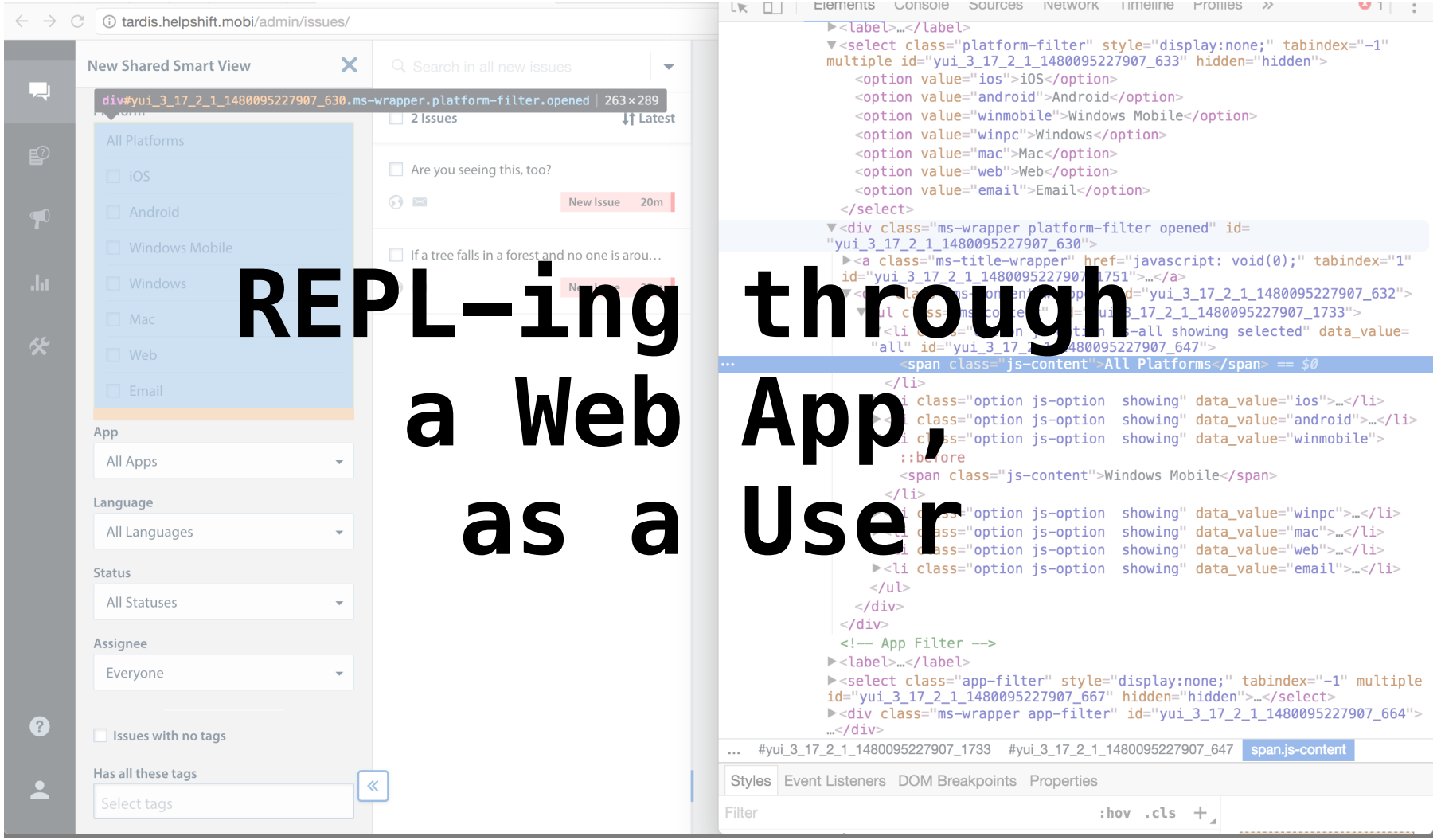
Object–Functional?

- A combination of
  - OO ideas, and
  - FP ideas
- For OO people
  - Me going from FP -> OO
  - Clues and References to model things in Clojure
- For FP people
  - Clojure-specific “System” / “Design” things … later

A live example

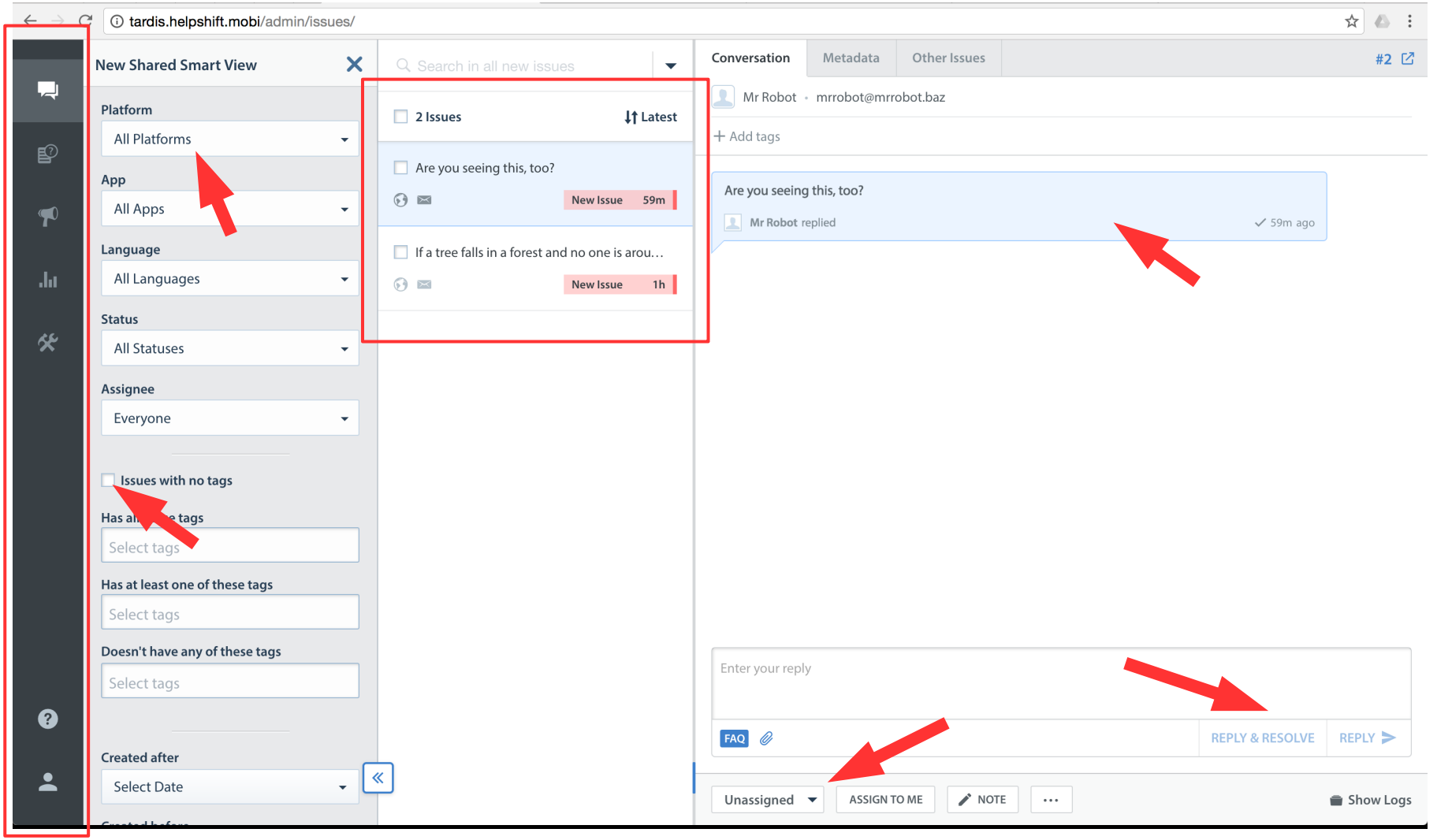
- Automating User Interfaces
  - Web App + Selenium Webdriver
- Develop intuition for “OO” in FP-land through the eyes of
  - Test framework author
    - Vs
  - Test framework users

Into the minefield...

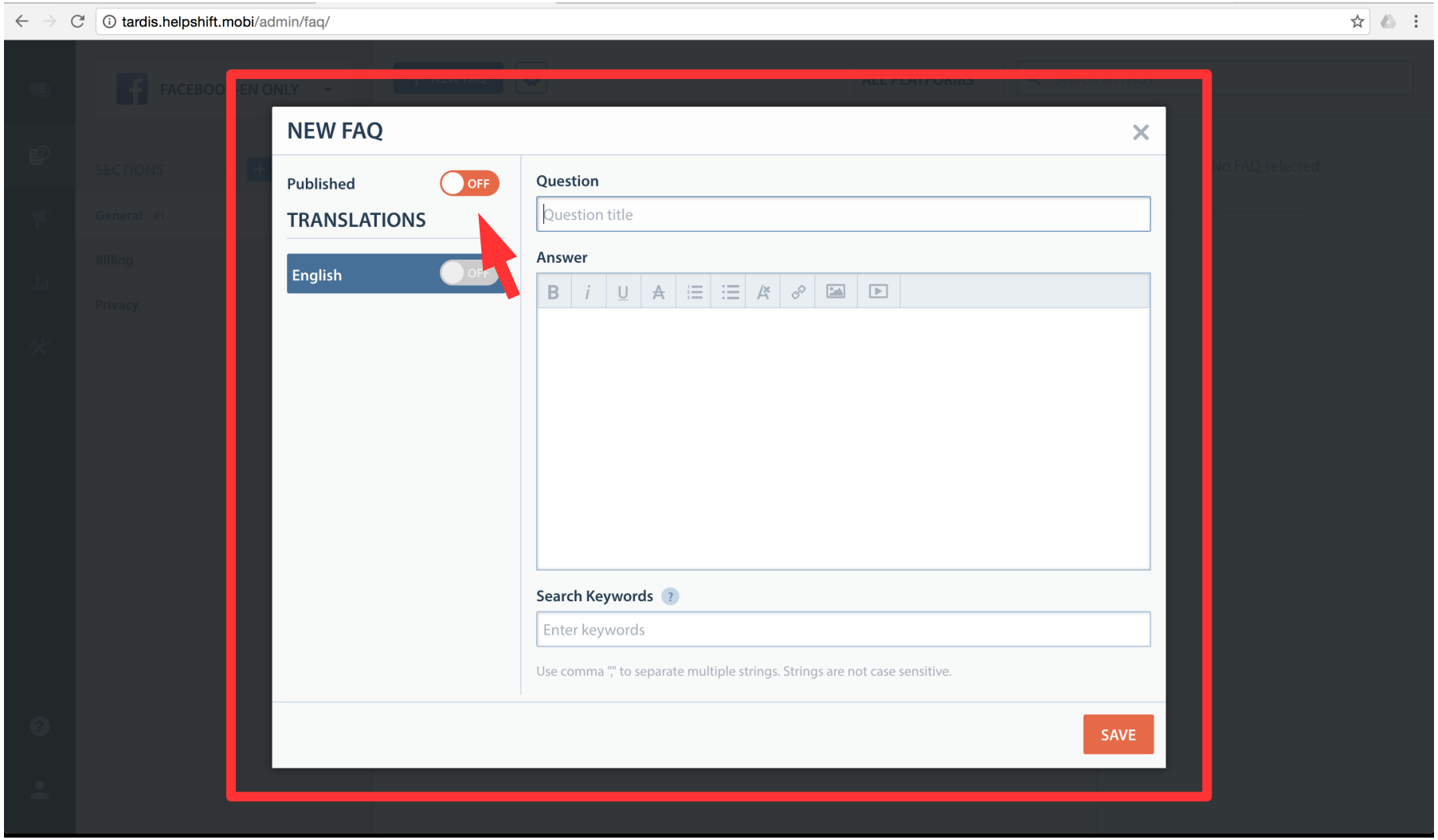


REPL-ing through a Web App, as a User

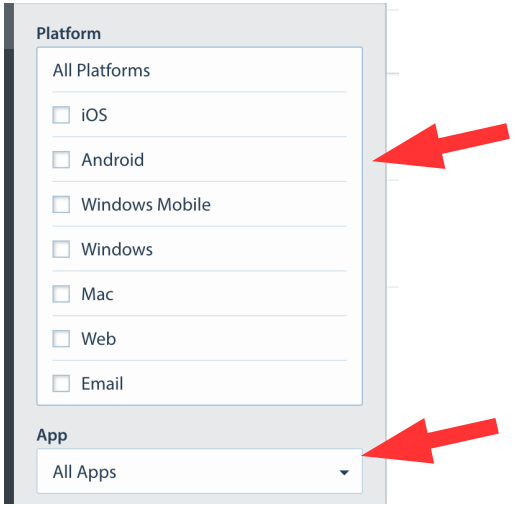
“Page Objects” (1/2)



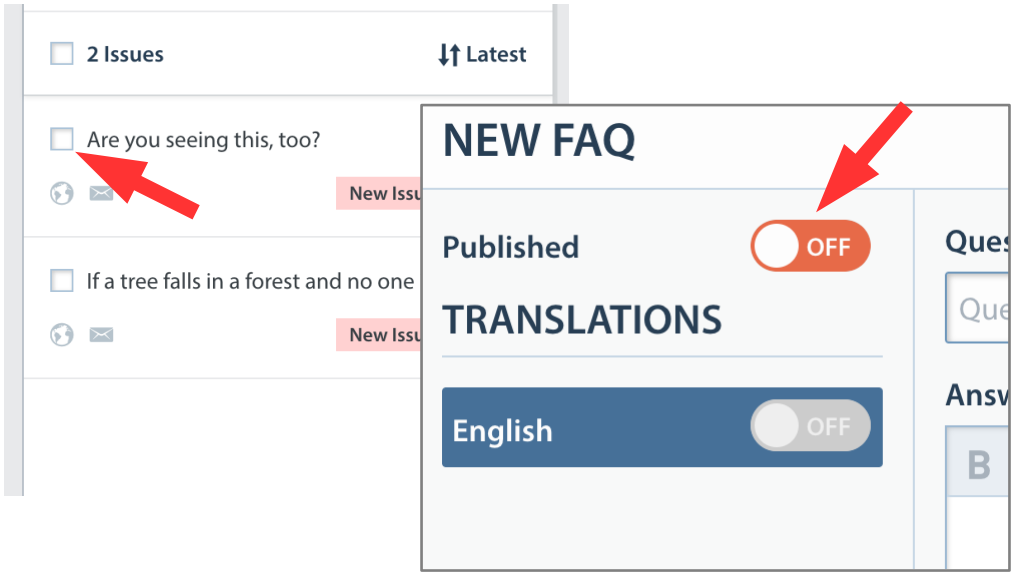
“Page Objects” (2/2)



Language of the Domain



- Dropdowns
  - exists? visible?
  - open, close, (open?)
  - select, deselect, get-value



- Checkboxes, Switches
  - exists? visible?
  - select, deselect, (selected?)
  - get-value

Cross-cutting concerns

	open	close	open?	select	deselect	...
Dropdown						
Checkbox						
Switch						
Modal						
SearchList						
...						
...						

“Expression Problem”

	List.add	List.get	List.clear	List.size
ArrayList				
LinkedList	Existing method implementations			
Stack				
Vector				
Your new class here	Your new method implmentations			

- OO: easy to add rows
  - ***new types*** extending a ***known interface***

	conj	nth	empty	count	
list					Your new function here
vector	Existing implementations				
map					
set					

- FP: easy to add cols
  - ***new functions*** that operate on ***existing types***

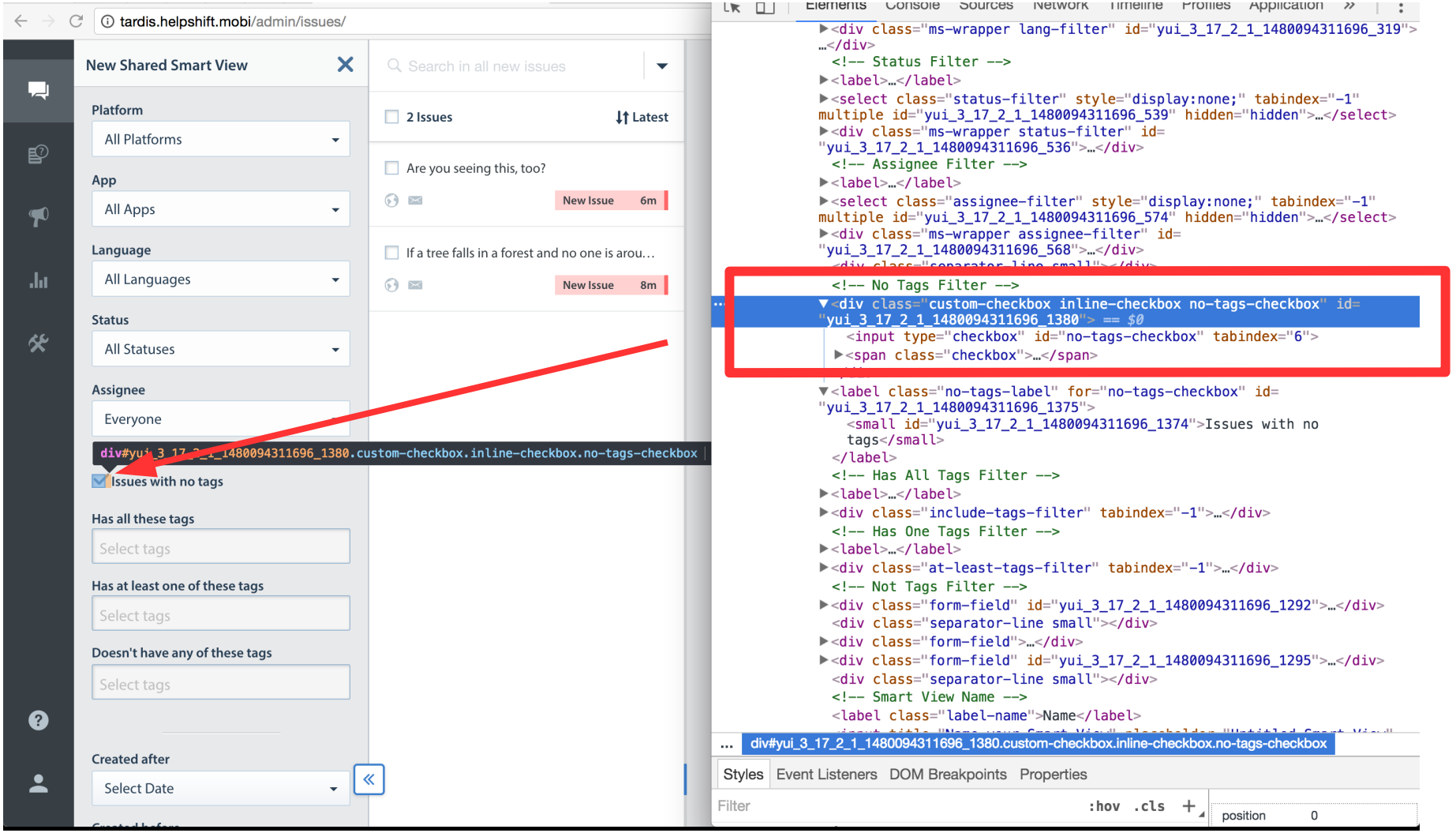
cf. http://www.ibm.com/developerworks/library/j-clojure-protocols/index.html

And it gets messier...

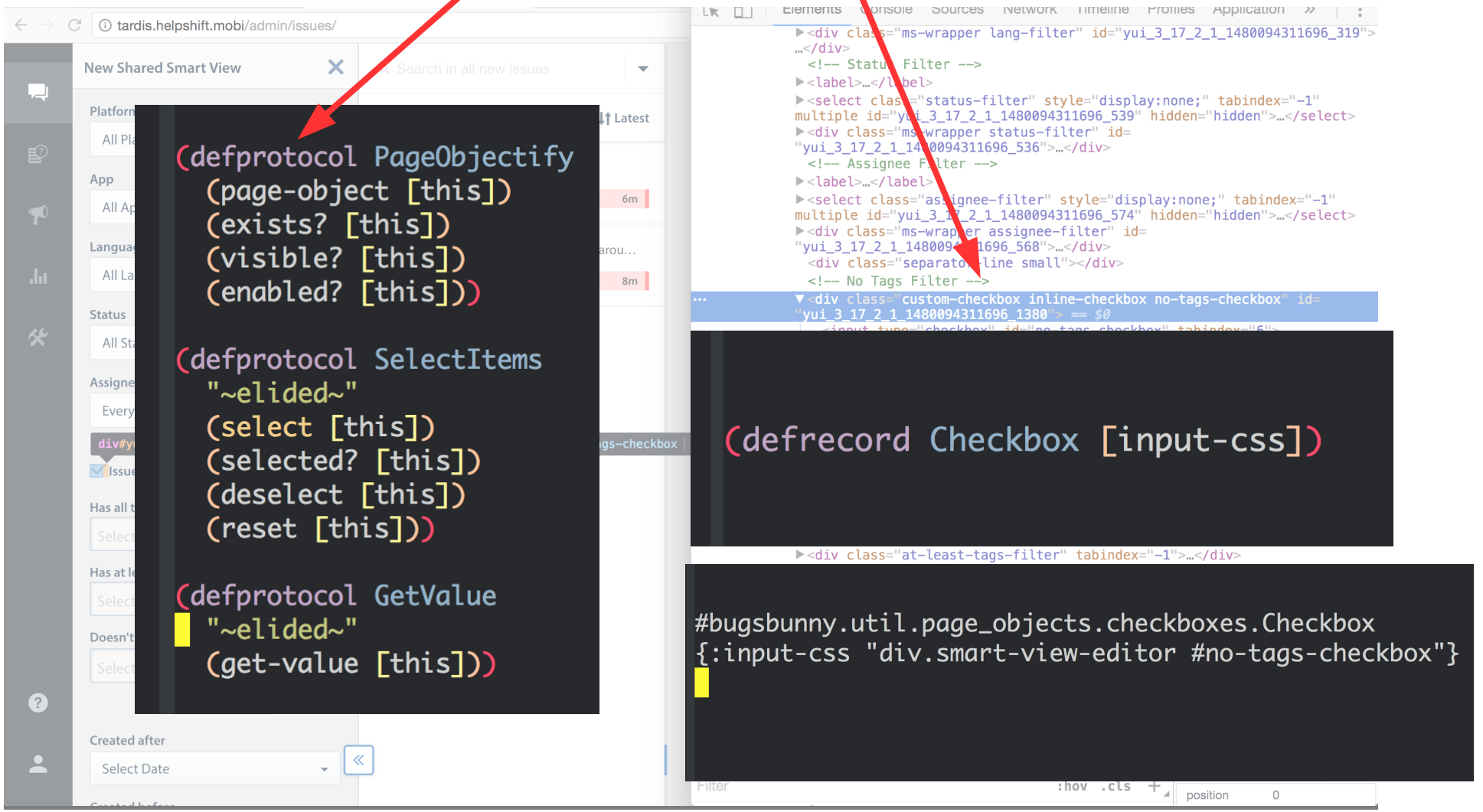
- Run-of-mill changes
  - CSS changes (tweaks / fixes)
  - Special Cases happen (and variants of special cases)
  - HTML signature change (refactor design)
- ***Everything*** changes (YUI -> React)
- UI = Mutable Soup (incidental complexity)
- Team's needs differ
  - Maintainers v/s Test writers v/s Product

How to Solve It?

Modelling a “Page Object”



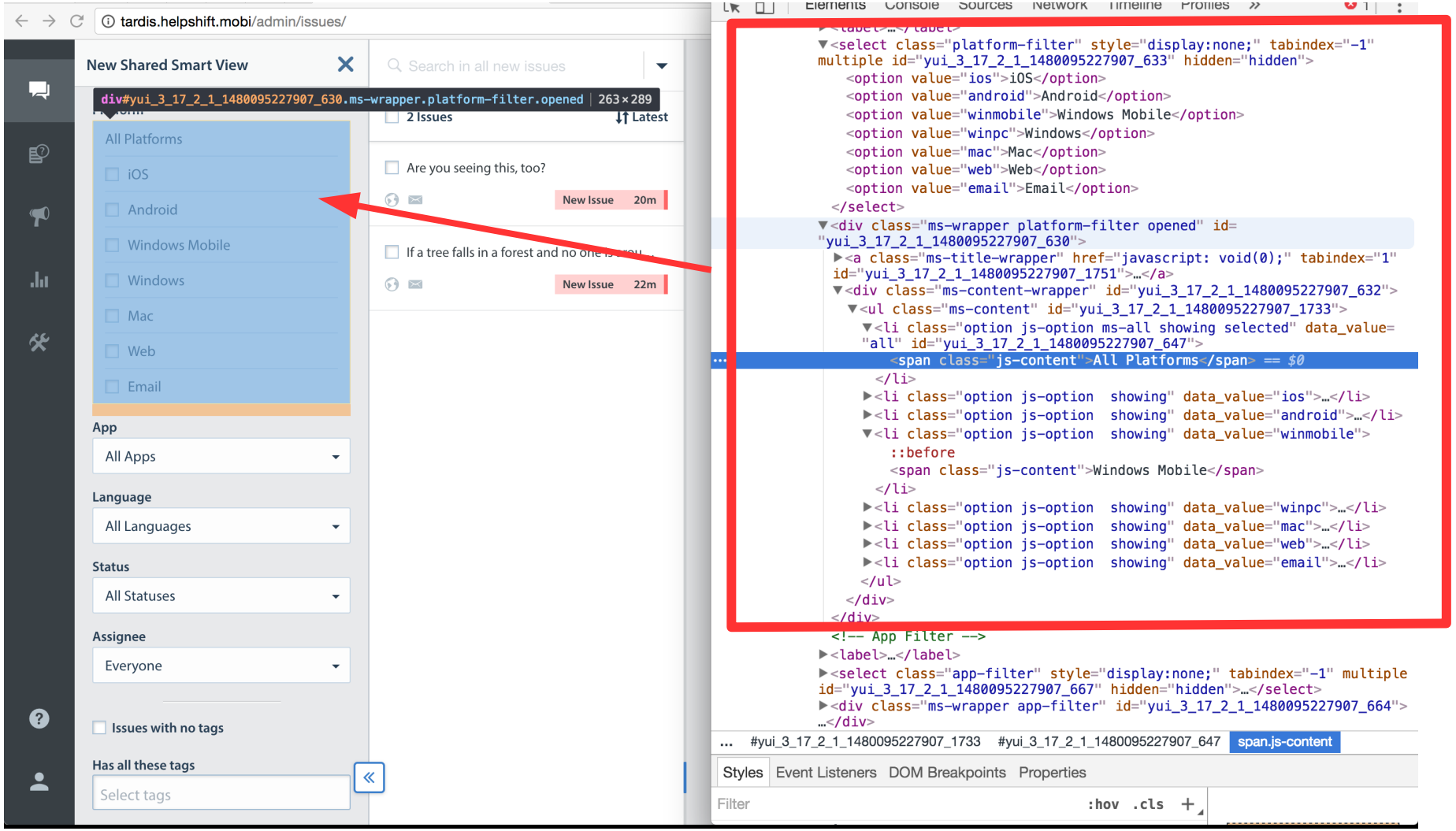
Modelling a “Page Object”
Behaviour + Structure



What are Clojure Protocols (and Records), and Why?

- Steal *all* of Stu Halloway's ***excellent*** talk
  - https://gotocon.com/aarhus-2010/presentation/Clojure%20Protocols%20are%20*not*%20Interfaces!
- Protocols solve the “expression problem”
  - Generic fns
    – arbitrary dispatch
    + speed + grouping
- Records specify information
  - Named type + kwd access + constructors + immutability + RASYDHT
  - e.g. Page Object structure

Down the REPL rabbit hole...



“Design”-y “System”-y things

- Safety things
  - Invariants, Sane Defaults, & Enforceable Conventions
- People things
  - Separate concerns, Live REPL help, Debugging (printable)
- De-risking development / workflow
  - Known knowns : Design by hammock (in any lang :-)
  - Known unknowns : Design by discovery (fast iteration)
  - Unknown unknowns : Hedge via judicious use of built-in abstractions (remember RASYDHT?)

Learning Material (1/2)

- Talks / Tutorials
  - Stuart Halloway on Clojure Protocols
    - goto conf 2010 : *http://gotocon.com/dl/jaoo-aarhus-2010/slides/StuartHalloway_ClojureProtocolsArenotInterfaces.pdf*
  - Sean Devlin's demo on Clojure Protocols
    - clojure conj 2010 : *https://www.youtube.com/watch?v%3DkQhOlWXXl2I*
  - Java.next: “Extension without Inheritance”
    - Pt1: *http://www.ibm.com/developerworks/library/j-jn5/index.html*
    - Pt2: *http://www.ibm.com/developerworks/library/j-jn6/index.html*
  - Michał Marczyk on Defrecord Deftype in Clojure(Script)
    - Clojure/West '16: *https://www.youtube.com/watch?v=RYGnLf_W3tk*

Learning Material (2/2)

- Books (I've used)
  - Programming Clojure
  - Joy of Clojure
  - Others: Brave Clojure,
- Codebases
  - Clojure itself: *e.g. clojure.core.protocols, clojure.data etc...*
  - Clj-webdriver: *https://github.com/semperos/clj-webdriver*
- REPL / reflection
  - (class <foo>), (supers <foo>), (clojure.reflect/reflect <foo>)

Ideas to use Protocols / Records (1/2)

- Javaland
  - Libs for expression-problem scenarios
    - Middleware / wrapper-ware / adapter-ware
    - Low-risk experiments / trials
- Clojureland
  - Stuart Sierra's 'component' for
    - app lifecycle mgmt.,
    - dependency injection in immutable world

Ideas to use Protocols / Records (2/2)

- Clojureland (contd...)
  - Mocks for test suites e.g.
    -
```
(reify MongoDataStore
       ;; actually do things in-memory
       (insert [_ doc])
       (update [_ query doc])
       (upsert [_ query doc])
       (delete [_ query])
       (commit [_])
       (stop [_]))
```
  - Etc...

So...

"Designing an 'Object-Functional' System with Clojure?"

- Design?
  - 'designare' (verb, Latin)
  - mark out, devise, choose / designate / appoint
- Object-Functional?
  - "A combination of OO *ideas* and FP *ideas*"
  - 'cause they're good together
- System?
  - 'systema' (Latin, and Greek)
  - an arrangement / organised whole / compounded of parts
- Clojure?
  - System design + problem-solving mojo

(((IN/Clojure)))

Nov 26, 2016

Λ

~ fin ~

@AdityaAthalye

helpshift

Extra...