

reagent-project / reagent

Watch 132


Star 1,728

Fork 130


[Code](#) [Issues 23](#) [Pull requests 13](#) [Wiki](#) [Pulse](#) [Graphs](#)

Warning: Reactive deref not supported in seq #18

New issue

 Open

 jonase opened this issue on Feb 19, 2014 · 3 comments




jonase commented on Feb 19, 2014

I see the warning "Reactive deref not supported in seq". Is this limitation fundamental to how the reagent atoms work? Or is it something not yet implemented?

In many cases it's easy to avoid the warning simply by lifting the deref out of the for loop. And if that doesn't work you can wrap your lazy-seq in a `doall`. I have yet to find a case where I can't get rid of the warning.

Would it be better to throw an exception instead of a warning? Or can the warning in some cases be safely ignored?



holmsand commented on Feb 19, 2014

Owner

Those are very good questions, in the sense that I'm not sure of the answers to them yet :)

First the basic problem: since seqs in general are lazy, they are typically not evaluated in the function where they are declared. So if you have something like

```
(def foo (atom ["hello"]))  
  
(defn bar []  
  [:div  
    (for [i (range 1)]  
      [:p (@foo i)])])
```

then `foo` would never be deref'ed when `bar` is run, but rather inside the `:p` component (or possibly the `:div`). Then obviously `bar` doesn't know that it needs to re-run when `foo` has changed, and general badness ensue.

This *could* be solved for most cases, by keeping track of the current "owner" component, and pass that along to children, so that when the seq is eventually evaluated, a dependency is added to, say, `bar`. But getting that right for *all* cases is kind of a bitch. :)

And I can't really think of a case where you *wouldn't* be better off lifting the deref out of the for loop. That is better for performance, and makes it more obvious what happens.

The exception vs. warning question is also a good one. `bar` would for example have been perfectly safe if it had looked like:

```
(defn bar []  
  @foo  
  [:div  
    (for [i (range 1)]  
      [:p (@foo i)])])
```

Then `bar` is re-rendered when `foo` changes, and everything works just fine. You just lose a tiny bit of performance from the extra deref. So maybe in this case it would have been a bit harsh to throw an

Labels

None yet


Milestone

No milestone

Assignee

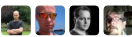
No one assigned

Notifications

 Subscribe

You're not receiving notifications from this thread.

4 participants



exception – but I'm not 100% sure about that...

There can be other cases as well, if (for example) you're perfectly ok with the seq not being updated when the atom changes. I actually had that very case in the color palette demo here:

<http://holmsand.github.io/reagent/news/reagent-is-async.html>

But even in that case, I would have been better off using explicit values, so the demo has now been changed – so in that case I appreciated the warning :)

So I'm still unsure if it is actually worth the extra complexity to allow derefs in seqs, and if it should be warning or exception when it happens. And since I'm not 100% sure, a warning seemed like the right thing to do.

Brilliant ideas are very much welcome. And maybe the warning could have better wording as well.

  holmsand referenced this issue on Feb 21, 2014

Maybe make native components reactive #12

🔒 Closed



Skinney commented on Aug 6, 2014

You can also prepend the 'for statement with 'doall

```
(defn bar []  
  [:div  
    (doall (for [i (range 1)]  
               [:p (@foo i)])))]
```



theophilusx commented on Oct 3, 2015

Could you just do

```
(into [:div]  
  (for [i (range 1)]  
    [:p (@foo i)]))
```



Write

Preview

 Styling with Markdown is supported

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Comment

