U08JL5H89 : However, it leaks memory very quickly.
U3QF0EM0E : yes <@U0702F2CE> thanks for pointing out `datum->syntax`. Here's another phase 1 example (still returns the weird result)```
#lang racket/base
(require (for-syntax racket/base))

(define-syntax (foo stx)
  (with-syntax ((x (datum->syntax stx (vector-immutable 1))))
    (syntax (syntax x))))

(immutable? (syntax-e (foo)))
```

U08JL5H89 : BUT, if I remove line 15 (the deallocator wrapper), it stops leaking memory.
U08JL5H89 : Does anyone have any suggestions?
U0702F2CE : <@U3QF0EM0E> here's the bug:
U08JL5H89 : (For the record, if I remove both alloc and dealloc, and just call the `av-frame-free` function by hand everything works fine.)
U0702F2CE : ```&gt; (immutable? (syntax-e (quote-syntax #(1))))
#f
```

U0702F2CE : note that `syntax->datum` does the right thing
U3NJS8H7C : leif: The short answer is that the finalization system used by `allocator` is not good enough (and wouldn't be good enough in most runtimes). Finalizers via `allocator` are run in a separate thread, so the program requires a combination of a GC and a context switch and another GC before some relevant memory can be reclaimed. Some solutions might involve limiting the allocator via `(sleep)` or forcing an occasional GC via `(collect-garbage)`.
U3NJS8H7C : I now see the rest of your original message, and I'm puzzled offhand that removing `#:wrap (deallocator)` helps, so I'll investigate a little more.
U08JL5H89 : <@U0702F2CE> and <@U3QF0EM0E> FWIW, I ran this test on Racket7, and got:```
leif@FATT ~/src/racket7/racket/bin (master) $ ./racket test.rkt
----
FAILURE
name: check-false
location: test.rkt:4:0
params: '(#t)
message: "(syntax-e (syntax (vector))) made immutable vector"

FAILURE
name: check-false
location: test.rkt:19:0
params: '(#t)
message: "(syntax-e (syntax (vector))) made immutable vector"

```

U08JL5H89 : So I'm pretty sure there is a bug here.
U3QF0EM0E : ^ this test = <http://pasterack.org/pastes/86496>
U3QF0EM0E : (point is, now the failures are consistent)
U3NJS8H7C : As far as I can tell, adding `#:wrap (deallocator)` slows down `av-frame-free` enough that the finalization thread moves more slowly than the allocation thread, so my original answer is still what I think is happening. Removing `#:wrap (deallocator)` might be a reasonable solution if `av-frame-free` is not to be called directly.
U08JL5H89 : Ah, okay, thanks makes a lot of sense. Thank you.
U08JL5H89 : That would also explain why writing my own finalizer (and a single thread that calls the will executors in a loop), doesn't have the same problem.
U07SJGB4H : If I register a will for a value exported by a module, can that will ever be executed?
U07SJGB4H : use case is a value used by test modules, wondering if it can be automatically cleaned up after the tests finish by relying on `raco test` somehow
U3NJS8H7C : <@U07SJGB4H> The namespace where the module is instantiated would have to become inaccessible,
```

including not being `current-namespace`
U07SJGB4H : <@U3NJS8H7C> if I `raco test -p mypackage`, which contains some test submodules that require a module providing a value with an attached will, would `raco test`'s shutdown process trigger the will execution? would running the tests in parallel with `-j` affect that?
U3NJS8H7C : <@U07SJGB4H> No, I don't think `raco test` will trigger the will execution. Either you're in `--direct` mode and it's all done in one namespace or the relevant place/process exits still holding its namespace
U07SJGB4H : drat
U629NGMAM : leif: Hi Leif!
U5KU1HNKY : Awwww
U5KU1HNKY : is there an addendum for the redex book anywhere? the syntax in the book is a bit crufty. not sure what else changed
U08JL5H89 : Hello
U0702F2CE : <@U3NP867S6> JFYI:
U0702F2CE : ```#hash(("semver" . (success test-fail))
("racketscript-extras" . (no-docs build-fail))
("wrap" . (install-conflict build-fail))
("css" . (success test-fail))
("racketscript" . (no-docs build-fail))
("turnstile" . (success test-fail))
("racketscript-compiler" . (no-docs build-fail)))
```

U0702F2CE : those are the current build regressions
U0702F2CE : I believe that `racketscript` and `racketscript-compiler` are <https://github.com/racket/typed-racket/issues/579>
U0702F2CE : `wrap` is <https://github.com/racket/typed-racket/issues/581>
U0702F2CE : `semver` is <https://github.com/racket/rackunit/issues/60>
U0702F2CE : `turnstile` is a timeout in the test suite
U0702F2CE : `css` I don't understand the test failure
U0702F2CE : also, `turnstile` passes on Travis on HEAD so we probably don't need to worry about it
U07SJGB4H : <@U0702F2CE> the rackunit issue is caused by one of my changes to the typed rackunit code in `typed-racket-more`. On a semi-related note, what would you think of moving the typed rackunit wrapper into the `racket/rackunit` repo?
U3QF0EM0E : zenspider: no I don't think there's any "readers guide" that connects the book to "modern day redex" :slightly_smiling_face:
```