U051SS2EU : right, it's a parser machinery basically
U051SS2EU : we're at the point in this convo where I'd probably have to read more code to answer more questions though
U5ZAJ15P0 : Luckily for you I don't have more questions at the moment :smile:
U051SS2EU : like whether there's an actual unquote method somewhere in Compiler.java or if it's entirely a figment of syntax-quote
U5ZAJ15P0 : For now I think I am contempt with the answer that clojure.core/unquote is not a function, but something that has a special interpretation within a syntax quote
U051SS2EU : there's this, but I am not totally sure what it means - UNQUOTE used to be defined as a special symbol but now it is commented out <https://github.com/clojure/clojure/blob/master/src/jvm/clojure/lang/Compiler.java#L70>
U1KK3BW3W : Ah, to be more specific, I am using `lein eastwood '{:add-linters [:unused-namespaces] :namespaces [:source-paths] :exclude-linters [:unlimited-use]}'` as my command, and I get an error for a namespaced spec (so you might be right that a namespaced keyword would not create problems, sorry!).
U060FKQPN : unquote never makes it to the compiler
U1KK3BW3W : ```== Eastwood 0.2.4 Clojure 1.9.0-alpha17 JVM 1.8.0_121Exception in thread "main" clojure.lang.ExceptionInfo: Invalid token: ::spec-ns/spec-name {:type :reader-exception}```

U060FKQPN : it's handled by the reader earlier than that
U1KK3BW3W : ```(ns acme.spec-ns (:require [clojure.spec.alpha :as s]))

(s/def ::spec-name string?)
```

U5ZAJ15P0 : <@U060FKQPN> could you point at where exactly it is handled?
U060FKQPN : <https://github.com/clojure/clojure/blob/master/src/jvm/clojure/lang/LispReader.java#L986-L990>
U060FKQPN : <https://github.com/clojure/clojure/blob/master/src/jvm/clojure/lang/LispReader.java#L1040-L1044>
U060FKQPN : from within the implementation of syntax quote
U060FKQPN : the tl;dr is that when the compiler sees `~foo` it expands into `(clojure.core/unquote foo)`
U5ZAJ15P0 : <@U060FKQPN> ah, so if it's an unquote form it simply gets rid of the "unquote" and "quote" bits at reading time?
U060FKQPN : then syntax-quote walks its expression and expands all the unquoted forms
U060FKQPN : you can see how that gets transformed by quoting a syntax-quoted expression
U060FKQPN : ```user=&amp;gt; '`(~a)(clojure.core/seq (clojure.core/concat (clojure.core/list a)))
```

U5ZAJ15P0 : so the reader would transform `(clojure.core/unquote 'foo)` into `foo`?
U060FKQPN : it's more complex than that
U5ZAJ15P0 : oh
U060FKQPN : because of splicing
U5ZAJ15P0 : if there was no splicing then it would be this simple?
U060FKQPN : more or less, yes
U060FKQPN : but because of splicing, it has to wrap every subform that is unquoted in a list & concat them all
U060FKQPN : so splicing becomes just like unquote w/o the wrapping list
U060FKQPN : ```user=> '`(~@a)(clojure.core/seq (clojure.core/concat a))
user=>
```
```