U2TCUSM2R : is it enough to just unquote the expression? that works
U2TCUSM2R : but same error when calling the macro
U051SS2EU : the next problem is that decls isn't going to be a valid data literal
U051SS2EU : so it needs to be escaped or quoted in some manner
U2TCUSM2R : hmm. it worked fine without the metadata
U051SS2EU : right but the decls are a data literal in the metadata - so they need to be a valid one
U051SS2EU : ```+user=&gt; (defmacro defn [name &amp; decls] `(def ~(with-meta name {:ast (cons 'quote decls)}) (fn ~decls)))#'user/defn
+user=&gt; (defn baz [])
#'user/baz
+user=&gt; (meta #'baz)
{:ast [], :line 19, :column 1, :file "NO_SOURCE_PATH", :name baz, :ns #object[clojure.lang.Namespace 0x373ebf74 "user"]}
user=&gt;
```

U2TCUSM2R : oh that one
U2TCUSM2R : i confused the error message
U051SS2EU : wait I think cons ins the wrong way to do that... checking
U051SS2EU : yeah, my bad ```+user=&gt; (defn baz ([]) ([_]))CompilerException clojure.lang.ExceptionInfo: Wrong number of args (2) passed to quote {:form (quote ([]) ([_]))}, compiling:(NO_SOURCE_PATH:21:1)
```

U051SS2EU : this fixes an issue with multi arities in the original too ```+user=&gt; (defmacro defn [name &amp; decls] `(def ~(with-meta name {:ast (cons 'quote (list decls))}) (fn ~@decls)))#'user/defn
+user=&gt; (defn baz ([]) ([_]))
#'user/baz
+user=&gt; (meta #'baz)
{:ast (([]) ([_])), :line 25, :column 1, :file "NO_SOURCE_PATH", :name baz, :ns #object[clojure.lang.Namespace 0x373ebf74 "user"]}
```

U051SS2EU : there's probably a better way to rewrite`(cons 'quote (list decls))` - that's super ugly
U2TCUSM2R : it works if i just do `(list 'quote decls)`
U051SS2EU : oh, right, much nicer
U2TCUSM2R : wow, thanks for explaining that to me as always
U2TCUSM2R : definitely would not have picked up on that on my own
U051SS2EU : yeah, there's an art to this stuff, and seeing simple examples makes a difference in learning it
U2TCUSM2R : tbh i never cared much about macros before the project i'm using this for
U2TCUSM2R : it's funny because in scheme i'm pretty sure i'd do some fancy transformation like closure conversion, but that's not possible since clojure is JIT compiled at the function level. but otoh, i don't think i could do it this way with scheme-style macros
U2TCUSM2R : now i get to play around with techniques for parsing the asts. i'll check out the muir library, but suspect i'll end up having to hack away at it myself
```