U11BV7MTK : ```employee-resizer.core&gt; (char 160)
\
employee-resizer.core&gt; \
\space
```

U0NCTKEV8 : <https://dev.clojure.org/jira/browse/CLJ-2207>
U5ZAJ15P0 : it had been a while since I debugged a unicode issue; almost forgot how fun it is
U5ZAJ15P0 : :kappa:
U11BV7MTK : apparently atom will enter a non-breaking space with alt-space
U11BV7MTK : probably pretty easy to inadvertently hit
U5ZAJ15P0 : yep, I most definitely inadvertently hit that
U5ZAJ15P0 : If Clojure's reader threw an error saying "unknown character at position X" it would have been easy/easier to debug, but it considered it as part of the symbol
U0NCTKEV8 : that actually may be just be a bug in the reader
U5ZAJ15P0 : Now I know how to confuse the hell out of people though
U5ZAJ15P0 : ```wef-backend.core=&gt; (def     42)
#'wef-backend.core/
wef-backend.core=&gt;
42
wef-backend.core=&gt;
```

U5ZAJ15P0 : this works
U5ZAJ15P0 : this too:
U5ZAJ15P0 : ```wef-backend.core=&gt; (def hello world this is a long variable 99)
#'wef-backend.core/hello world this is a long variable
wef-backend.core=&gt;
```

U5ZAJ15P0 : eh
U5JUDH2UE : What is a fast way to do this without the repeated nested pointer?```
(-&gt; state
    (assoc-in [:a :b :b/field0] "")
    (assoc-in [:a :b :b/field1] "")
    (assoc-in [:a :b :b/field2] ""))
```

U0CM1QURZ : ```;; maybe?
(assoc-in [:a :b] {:b/field1 "" :b/field1 "" :b/field2 ""})
```

U050MP39D : fast as in performance?
U5JUDH2UE : Fast as in, using standard library. I don't want to add any external libs or anything.
U5JUDH2UE : Performance isn't a concern.
U5JUDH2UE : Sorry, I shouldn't be using that work. :stuck_out_tongue:
U050MP39D : (merge state {:b/field1 "" :b/field1 "" :b/field2 ""})
U050MP39D : err
```