

Carp

A Language for the 21st Century

Veit Heller

October 23, 2018

Localhost | Recurse Center

- RC Summer 1 2017
- Carp standard library maintainer
- ... But not the creator!
- Secretly a turtle

man carp

- a Lisp-1
- type-inferred
- borrow-checked
- compiles to (somewhat readable) C
- for realtime applications

- a Lisp-1
- type-inferred \Rightarrow statically typed, at no extra charge
- borrow-checked \Rightarrow no GC, at no extra charge
- compiles to C
- for realtime applications

- Haskell implements a Hindley-Milner type system and inference
 - ⇒ You don't have to spell types out!
- Rust implements borrow checking
 - ⇒ You don't have to manually manage memory, even without a GC!

Let's put those things together (after simplifying) and rejoice!

⇒ Also add some Lisp macro goodness and a near-seamless C FFI for good measure!

source carp

```
; (type f)
; f : (Fn [(Ref (Array a)), Int, Int] a)
(defn f [x y z]
  @(Array.nth x (* y z)))
```

Listing 1: A silly Carp function

```
(deftype (AssocArray a b) [lst (Array (Pair a b))])
```

Listing 2: An associative array type, simplified.

```
(defmodule AssocArray
  (defn put [a k v]
    (let [list (lst a)
          pair (Pair k v)
          new-list (Array.push-back list pair)]
      (set-lst a new-list)))
)
```

Listing 3: A module for the associative array.

touch anima.carp

anima.carp is a simple animation framework based on Processing.

open demo.live

Anima uses the C FFI, a simple DSL, and macros to do its magic.

```
touch anima.carp
```

```
(local-include "../core/SDLHelper.h")  
(add-cflag "`sdl2-config --cflags --libs`")  
  
(defmodule SDL  
  (register init (Fn [Int] ()))  
  (register delay (Fn [Int] ()) "SDL_Delay")  
  
  ; ...  
)
```

Listing 4: Wrapping SDL.


```
touch anima.carp
```

```
(defmodule Anima  
  (defn line [rend ax ay bx by]  
    (SDL.render-draw-line rend ax ay bx by))  
  
  ; ...  
)
```

Listing 5: Writing a DSL.

```
(defdynamic setter [f r color cs]
  (if (= (length cs) 0)
    (list f r color color color 255)
    (if (= (length cs) 2)
      (list f r color (car cs) (cadr cs) 255)
      (list f r color (car cs) (cadr cs) (caddr cs))))))

(defmacro color [r color :rest cs]
  (setter 'set-color r color cs))
```

Listing 6: Writing a macro.

reboot c.ffi

Do you have to write all of the bindings by hand?

```
reboot c.ffi
```

Of course not!

```
reboot c.ffi
```

There's a tool for that!

open demo.live.2

exit

Carp is early stage software.

- ⇒ Small community, few packages
- ⇒ We're less than a handful of maintainers
- ⇒ Insufficient documentation
- ⇒ May change under your feet
- ⇒ May blow up in your face!

We're approaching the first stable release (0.3)

- Github: <https://github.com/carp-lang/carp>
- Erik: <https://github.com/eriksvedang>
- Chat: <https://gitter.im/carp-lang/carp>
- Docs & Blogs: <https://blog.veitheller.de> (sorry about that)
- Slides: https://github.com/hellerve/carp_talks
- This talk, but different, shorter, and at clojuTRE:
<https://www.youtube.com/watch?v=BQeG6fXMk28>

Thank you!

Questions?

Slides at https://github.com/hellerve/carp_talks