Carp

A Language for the 21st Century

Veit Heller

September 21, 2018

Port Zero

man carp

```
man carp
```

- a Lisp-1
- type-inferred
- borrow-checked
- compiles to C
- for realtime applications

```
man carp
```

- 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

```
whence -v carp
```

- 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

```
touch anima.carp
```

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

```
touch anima.carp
```

It 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.

```
touch anima.carp
```

```
(defmacro defsketch [name w h setup draw]
  (list 'defn 'main (array)
    ; ...
    (list setup 'rend)
    (list 'while true
      (list draw 'rend))))
```

Listing 6: Writing a macro.

open demo.live

cmp clojure carp

Carp	Clojure
Statically typed	Dynamically typed
Compiled	Kind of compiled?
Works with C	Works with Java (and JS)
Modules	Namespaces
Compile-time metaprogramming	full metaprogramming
Interfaces	Protocols

exit

trap

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)

trap

- 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

Thank you!

Questions?

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