Carp

A Programming Language for the 21st Century

Veit Heller
Port Zero

July 12, 2019

Prologue

Obligatory Starter

I didn't build Carp.

I maintain parts of it, but all credit goes to Erik Svedäng (hi, Erik!).

This talk is not endorsed by him, and all opinions are my own.

Questions & Comments

This is a chess-timer talk. Ask questions at any time.

I can't promise that I'll be able to answer all of them.

I probably won't get through my material.

I. Syntax & other trivialities

Koalas

Let's talk koalas.

Koalas

Koalas are endangered!

Consider donating to the Australian Koala Foundation.
https://www.savethekoala.com/

What?

Carp's syntax is that of Lisp, parentheses and all.

Why?

The goal is homoiconicity or “code as a data structure”.

Carp code

```
; (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

Carp code

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

Listing 2: An associative array type, simplified.

II. Semantics & meaning

The goods

We have (non-hygienic) macros, type inference, and a borrow checker.

The emphasis of the design is on simplicity while being pragmatic.

What we're still tuning

We're working on

- good autogenerated documentation,
- dependency management,
- lifetimes, and
- tooling in general.

Why?

There's a vast amount of information flowing through the compiler.

Most of it is implicit.

Reducing cognitive load is useful, but explorability is key.

Why?

Everything should be inferred if possible, and exposed to our tools to help make our lifes easier.

Why?

I was fortunate enough to use Pharo professionally last year.

It's really that good.

What?

Macros are interesting abstractions.

I want to be able to explore their before, during, and after.

What?

Type inference is useful.

Most types are either trivial or painful to type, but easy to reason about.

I still want to know about the types occasionally. Especially when there was a misunderstanding.

What?

I don't want to deal with memory allocation.

I don't want my program to pause randomly.

And I want to know when things get deallocated because their scope ends.

The big reveal

We don't have any tools yet.

The big reveal

We've built a foundation, and released it.

Now it's time for us to get to work on interesting things.

III. Recap

Why am I here?

Designing a language is about affordances.

Macros?

Macros give you a unique ability to abstract.

They also blow up all the time, and leave a mess.

Typing?

Static types give you explorability and ahead-of-time guarantees.

Dynamic types give you velocity and malleability.

Memory management?

Manual memory management gives you memory layout control.

Garbage collection gives you (better) memory safety.

Borrow checking gives you safety without runtime pauses.

And Carp?

Carp pushes as much work as possible into the compiler.

Hopefully tools can pull the results of that work out again.

Caveat

Carp is early stage software.

- $\Rightarrow$ We're a small community with few packages.
- $\Rightarrow$ We're less than a handful of maintainers.
- $\Rightarrow$ Documentation is insufficient.
- $\Rightarrow$ It may change under your feet.
- $\Rightarrow$ It may blow up in your face!

Our first stable version—0.3.0—was released!

References

- Github: https://github.com/carp-lang/carp
- Erik: https://github.com/eriksvedang
- Chat: https://gitter.im/carp-lang/carp
- Docs: http://carp-lang.github.io/Carp/core/
- Blog: https://blog.veitheller.de
- This talk, but different, shorter, and at clojuTRE: https://youtu.be/BQeG6fXMk28
- Carp 0.3.0: https://github.com/carp-lang/Carp/releases/

Thank you!

Questions?

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