Introduction to ClojureScript and Functional Programming

200ok GmbH

Alain M. Lafon, 24.07.2018

alain@200ok.ch

Outline

Introduction

About me

Alain M. Lafon

CV

- CEO 200ok GmbH
- CEO QuickShift GmbH
- Lecturer at ZHAW
- Zen Monk, runs the Lambda Zen Temple

Contact

alain@200ok.ch

Questions

- I'll cover a lot of ground in this talk
- However, this is a lightning talk

- Please ask questions at any time, when something is unclear or you're simply curious

Before we begin

Any LISPers in the room?

Right on!

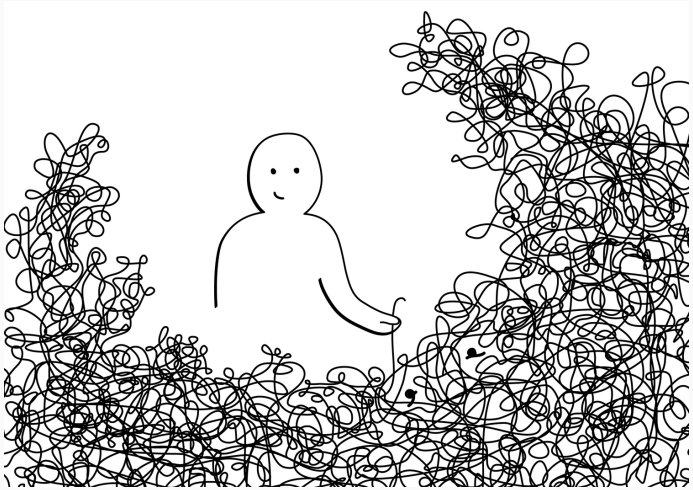
How did you get started with LISP?

- Did some of you start using LISP because of Clojure?

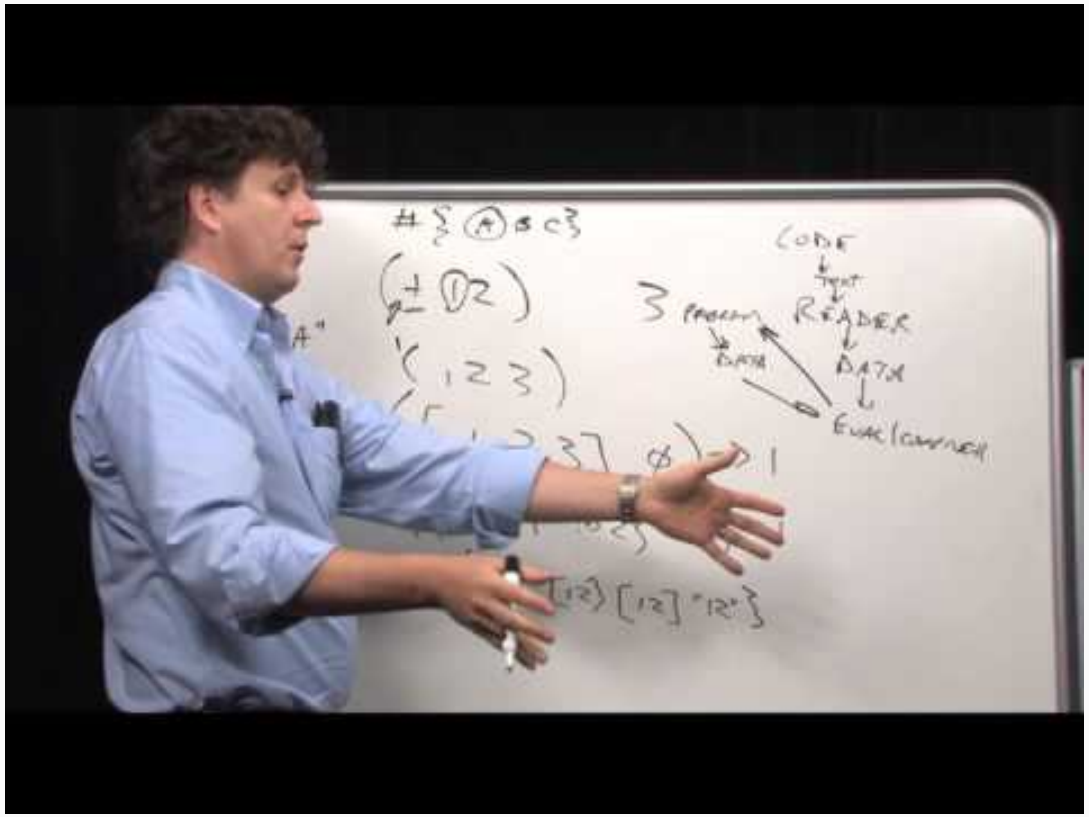
Complexity, Functional Programming and Clojure

Complexity

- The complexity of software is growing at an exponential rate
- The biggest problem is the growing complexity of dynamic state which makes it hard to reason about a system



"State - You're doing it wrong!"



Rich Hickey, creator of the Clojure

Quote Source: `http://clojure-log.n01se.net/date/2008-06-26.html#11:10a`

Incidental complexity

State is hard. And on top, we often use OOP and mutable state to make things worse. It is very easy to add incidental complexity to your application. Just look at this very normal code which probably doesn't look complex at all:

```
var x = 1;
var y = x + 1;
y == 2 == true;
x = 2;
y == 3 == false;
```

Due to mutable state, `y` was not updated after `x` has been changed. This makes reasoning about the current state and logic of the application hard.

How to design UI

The UI layer is most predictable when it is described as a pure function of the application's state.

Intro Clojure

Some of the benefits of Clojure/ClojureScript are:

- It's a Lisp! (Code as Data)
- Functional language with immutable persistent data structures
- Almost no syntax
- Amazing tooling, Editor integration, performance
- Use the same language on back-end and front-end
- Uses Google Closure to optimize your code and included libraries
- Uncle Bob believes it to be the last programming language

Syntax Example

Syntax example (counting clicks) written in Reagent which builds on top of ReactJS.

```
(ns example
  (:require [reagent.core :as r]))
(defonce click-count (r/atom 0))

(defn counting-component []
  [:div
   "Button has been clicked" @click-count " times"
   [:input {:type "button" :value "Click me!"
            :on-click #(swap! click-count inc)}]])
```

Live Demo

Live Coding - what can go wrong?

Resources

Talk tax

Slides and minimal example app

Will be uploaded later to `https://200ok.ch`

Watching / Reading List

Video

- Simple made Easy (Rich Hickey): `https://www.infoq.com/presentations/Simple-Made-Easy`
- Figwheel Demo (Bruce Hauman): `https://www.youtube.com/watch?v=KZjFVdU8VLI`

To read

- Clojure for the Brave and True: `http://www.braveclojure.com/`
- `https://200ok.ch/category/clojure.html`
- 200ok.ch/atom.xml

Jobs

In case you're searching for a job, talk or write to me!

Questions