The Strengths of Clojurescript

Eric Normand

PurelyFunctional.tv

Clojure History

- Created by Rich Hickey
- Released in 2007
- Lisp for the JVM
- Used at many companies
  - Amazon, PayPal, Ebay, Citibank, Walmart

ClojureScript History

- Created by Rich Hickey
- Released in 2011
- Project led by David Nolen
- Clojure transpiled to JavaScript
- Runs in Browser / Node / JavaScriptCore / etc.

What is Clojure?

- Functional Programming
- Functions
- Data

What is Clojure?

- Immutable Data Structures
- Share structure
- Literal representations

What is Clojure?

- Extensible language
- Macros
- DSL

What is Clojure?

- Hosted
- Rely on host VM features
- Good interoperation

What is Clojure?

- Dynamic
- Dynamic types
- Live runtime

function to call

first argument

```
(println “Hello, World!”)
```

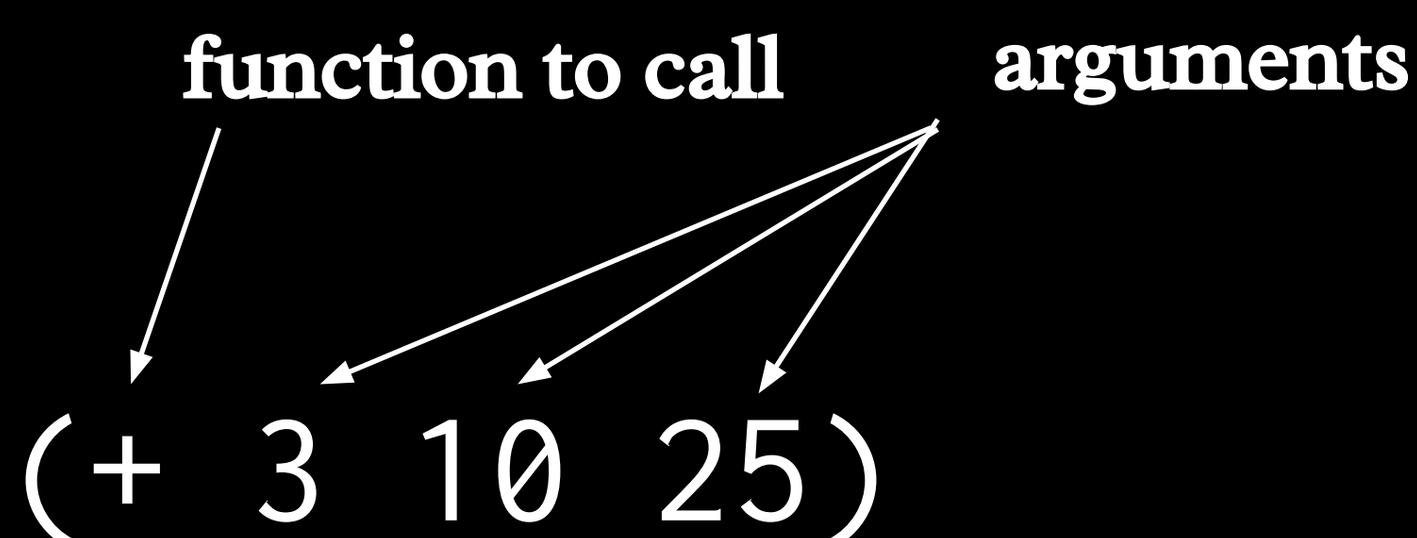
(JavaScript/Java)

function to call

first argument

```
println(“Hello, World!”)
```

Very regular syntax


function to call
arguments
(+ 3 10 25)

```
(if (pos? x)
(println “Positive”)
(println “Negative”))
```

loop through each item in list

```
(doseq [item list]
  (db/save item))
```

define a global variable

name of global variable

value of global variable

```
(def user {:id 345
           :username “eric”
           …})
```

Immutable hash map

keyword

define a global variable

name of global variable

value of global variable

```
(def admin-user-ids [34 22 56 776])
```

Immutable vector (array)

define a named function

name of function

argument list

body of function

```
(defn fetch-user [user-id]
  (get user-database user-id :not-found))
```

```
(defn fetch-admin-users []
  (map fetch-user admin-user-ids))
```

`map` **takes a function and a list**

Data Standard Library

- Sequential (lists and vectors)
  - map/filter/reduce
- Associative (vectors and hashmaps)
  - assoc/dissoc/get
- All collections
  - conj/disj/seq

JavaScript interoperation

- ClojureScript functions => JavaScript functions
- Strings
- Numbers
- Easily create JS objects and JS arrays
- Refer to JS environment (`js/window`)

Frontend Programming with Re-frame

What is Re-frame?

- Frontend framework
- Built on React.js
- Build functional components
- Manage state
- Manage effects

Hiccup

- DSL for embedding HTML in Clojure

```
[:div {:class “container”
       :style {:margin-left “10%”}}
  [:span “Hello, World!”]]
```

```
<div class=“container”
     style=“margin-left:10%;”>
  <span>Hello, World!</span>
</div>
```

Components

- DSL for embedding HTML in Clojure

```
(defn user-card [first-name last-name avatar]
  [:div {:style {:margin-left “10%”}}
    [:div first-name]
    [:div last-name]
    [:div
      [:img {:src avatar
             :alt (str first-name “ “ last-name)}]]])
```

Live Coding Demo

Styling HTML Live

Re-frame

- Database for application state
- Events for modifying database and other effects
- Subscriptions for reacting to state changes

Live Coding Demo

Adding a button that changes state

Using Existing Components

- CLJSJS - http://cljsjs.github.io/
  - Many pre-packaged components
  - Many React components on NPM
    - https://react.rocks/
    - NPM Support coming very soon
- Easy to create custom components

Re-frame Performance

- React.js is fast
  - Optimizes DOM changes
- Re-frame uses `requestAnimationFrame`
  - Fewer re-paints
- Immutable data
  - Reduces # of React component updates

Browser REPL Demo

REPL, Debugging

DSL Anatomy

Parse

String => Abstract Syntax Tree

Interpret

Abstract Syntax Tree => Actions and Value

Clojure Reader

- Parsing step can be done by Clojure reader
  - Run `clojure.edn/read`
    - (or `cljs.reader/read-string`)
  - Embed literal data structures directly in Clojure code

Clojure Reader Demo

Interpreted Component Demo

Multi-step Wizard

Other syntaxes

- Instaparse
  - EBNF Grammars
- Write your own

Eric Normand

LispCast

Follow Eric on:

Eric Normand

@EricNormand

lispcast.com

eric@lispcast.com