Fun With Clojure

History

21st Century

January 1 2001

Clojure

Clojure first appeared in 2007

Lisp

Lisp was first specified in 1958

CSP

First described in 1978

JVM

JVM was introduced in 1994

JavaScript

Appeared in 1995

Clojure

21st Century Problems

Evolvable Systems

- Quickly add new features for customers
- Refactor with Confidence

Resilient

- Open System - Handles large amounts of inputs that you have no control over
- Low response times

Something that involves JavaScript

Our Journey

Agenda

— Specifying our problems

— Concurrency / Parallelism

— ClojureScript

(whoami)

```
{
 :name "Jearvon Dharrie"
 :day-to-day "Developer Advocate @ Comcast"
 :twitter "@jearvon"
 :repo "https://github.com/iamjarvo/qconnyc_2018"
}
```

Clojure

Just Enough Clojure

```
(function arg)
```

```
(+ 1 2)
```

```clojure
(ns qcon.spec.core
  (:require [clojure.spec.alpha :as s]))

(s/valid? string? "1")
```

clojure.spec

Spec Anatomy

```
(spec-function predicate value)
```

```clojure
(s/valid? int? 3000)
;; true
```

Predicate

```
(x) → boolean
```

Builtin Predicates

uri?	true?
even?	pos-int?
odd?	any?

Custom Predicates

```
(fn [value] (>= value 18))
```

Named Specs

```clojure
(s/def ::port int?)
```

```
(s/valid? ::port 3000)
```

Spec Registry

Spec Maps

```
(s/def ::config
  (s/keys :req-un [::port ::env]))
```

Compose Specs

```
(s/def ::valid-port-range
  #(and (> % 1023)
        (<= % 65535)))

(s/def ::port
  (s/and int? ::valid-port-range))
```

What Can You Do With Spec?

Validate

```clojure
(s/valid? ::port "3000")
;; false

(s/valid? ::port 3000)
;; true
```

Conform

```
(s/conform ::port 3000)
;; 3000
```

```clojure
(s/def ::building #(re-find #"[0-9]+" %))
(s/def ::address
  (s/cat
   :building-num ::building
   :street string?))


(s/conform ::address ["1701" "JFK Blvd"])
```

Explain

```clojure
(s/explain ::port "3000")
;; val: "3000" fails spec: :qcon.core/port predicate: int?
```

```clojure
(s/explain ::port 400)
;; val: 400 fails spec: :qcon.core/valid-port-range predicate: (and (> % 1023) (<= % 65535))
```

Doc

```
(doc ::port)

;; Spec
;;     (and int? :qcon.core/valid-port-range)
```

Together

Spec Functions

```
(s/fdef find-by-id
  :args (s/cat map? ::valid-id)
  :ret map?)
```

```clojure
(defn find-by-id
  [db id]
  (first
   (filter #(= id (:id %)) db)))
```

```
(doc ::find-by-id)
```

Exploring

```
(gen/sample (s/gen ::config))
```

Instrument

```
(stest/instrument `find-by-id)
```

Test Check

```
(stest/check `find-by-id)
```

Concurrency & Parallelism

The future belongs to languages that can automatically leverage more cores as they become available
— Clojure Applied

Host Platform

— Thread

— java.util.concurrent

Pure Functions

```
(defn add [x y]
 (+ x y))
```

Immutable Data

```
(assoc {:first "Jearvon"} :last "Dharrie")

(let [m {:first "Jearvon"}]
  (assoc m :last "Dharrie")
  m)
```

Atom

```
(def qcon (atom 0))

@qcon
;; 0

(swap! @qcon inc)
;; 1
```

pmap

```
(pmap expensive-call collection)
```

Futures

Reducers

core.async

https://en.wikipedia.org/wiki/Communicating_sequential_processes

chan

```
(chan)
```

Put/Take

> !!   < !!

go

Put/Take

```
>!  <!
```

Token Bucket Filter

A token is added to the bucket every 1/r seconds.

Token Bucket Filter

The bucket can hold at the most b tokens. If a token arrives when the bucket is full, it is discarded.

Token Bucket Filter

When a packet (network layer PDU) of n bytes arrives, n tokens are removed from the bucket, and the packet is sent to the network.

Token Bucket Filter

If fewer than n tokens are available, no tokens are removed from the bucket, and the packet is considered to be non-conformant.

ClojureScript

Targets

- Node.js
- The Browser
- Anywhere that JavaScript runs

Language

Tools

Clojure Goodies

- Immutable Data Structures
- clojure.spec!
- core.async
- Libraries

Share Code

CLJC - Reader Conditionals

```
#?(:cljs (defn upcase [s] (.toLowerCase s)))
#?(:clj (defn upcase [s] (clojure.string/upper-case s)))
```

React Wrappers

- Reframe
- Om
- React Native

Browser REPL

Interactive Development

Source Maps

Differences

https://clojurescript.org/about/differences

Some Points

- Learning the language is easy
- Learning how to do things is difficult
- I am always learning and being challenged
- Need to accept the Clojure way - Simple
- Compose libraries instead of frameworks

Conclusion

- Clojure supports simplicity, pure functions and immutable data
- clojure.spec helps you define and validate the shape of your data
- Concurrency and parallelism are first class citizens in Clojure
- Clojure compiles to JavaScript and allows you to take advantage of working in the browser and with nodejs

Thanks