Clojure

Me

- Tap
- More of backend guy. First involved frontend work with ClojureScript
- Java -> Ruby -> Ruby + Clojure
- @visibletrap
- Clojure in Thai
- 2009 - started working professionally
- 2012 - started playing with Clojure
- 2015 - started using Clojure professionally

Clojure

- General purpose programming language
- Hosted language (with good interop)
  - JVM (Clojure, 2007)
  - JavaScript (ClojureScript, 2011)
  - some others
- Functional, Lisp, data-oriented

```
(println "Hello," "World")
```

Data structures

Vector

```
["Orange" "Apple" "Microsoft"]
```

List

```
("dog" "cat" "rat")
```

Data structures

Map

```
{1 "Monday" 2 "Tuesday" 3 "Wednesday" 4 "Thursday"
 5 "Friday" "6" "Saturday" :7 "Sunday"}
```

Set

```
#{"Pen" "Apple" "Apple Pen"}
```

Persistent data structures

- Immutable
- Structural sharing
- Fast
- They are values. Similar to int, string
- Can drop down to use transient data strucutre for performance

Function

```
(defn say-hello [name]
  (println "Hello," name))
```

Interactive development with REPL

- Use it for everything
  - Run snippet of code
  - Run test
  - Debug
  - Start server
  - Connectable to browser / mobile devices
- Feedback is the key
- Lower friction trying out a snippet of code
- Debugging on production!!!

Data modeling

- Use maps
- Reuse tons of collection functions
- Easy to serialize
  - JSON
- Easy to debug

Example - HttpServletRequest

getAsyncContext, getAttribute, getAttributeNames,
getCharacterEncoding, getContentLength,
getContentType, getDispatcherType, getInputStream,
getLocalAddr, getLocale, getLocales, getLocalName,
getLocalPort, getParameter, getParameterMap,
getParameterNames, getParameterValues, getProtocol,
getReader, getRealPath, getRemoteAddr, getRemoteHost,
getRemotePort, getRequestDispatcher, getScheme,
getServerName, getServerPort, getServletContext,
isAsyncStarted, isAsyncSupported, isSecure,
removeAttribute, setAttribute, setCharacterEncoding,
startAsync, startAsync...
getHeader, getHeaderNames, getHeaders, getIntHeader...

- Clojure, Made simple https://www.youtube.com/watch?v=VSdnJDO-xdg

In Clojure, Just Use Maps

```
{:remote-addr "127.0.0.1",
 :scheme :http,
 :query-params {"somekey" "somevalue"},
 :form-params {},
 :request-method :get,
 :query-string "somekey=somevalue",
 :content-type nil,
 :uri "/foobaz",
 :server-name "localhost",
 :params {"somekey" "somevalue"},
 :headers
          {"accept-encoding" "gzip, deflate",
           "connection" "close",
           "user-agent" "Apache-HttpClient/4.1.2 (java 1.5)",
           "content-length" "0",
           "host" "localhost:8383"},
 :content-length 0,
 :server-port 8383,
 :character-encoding nil}
```

- Clojure, Made simple https://www.youtube.com/watch?v=VSdnJDO-xdg

How do we sum these?

```
[{:item1 1 :item2 2} {:item1 3 :item3 5} ...]
```

expect

```
{:item1 x :item2 y :item3 z}
```

Data modeling

- **There are tons of function like `merge-with` and `diff` ready for use**
- **Official ones are all in Clojure Cheatsheet**
- **Practice at https://www.4clojure.com/**

Encourage pure functions, minimize side effects

— There's no purity enforcement

Example of side effects

- Read / Write from to I/O (file, database, http)
- Print to console
- Input from keyboard. Button click
- Setter method

Separate side effects - ideas

- ETL job
- Form validations
- REST

Separate side effects - benefits

- Easy to develop
  - Refactor
- Enable reuse
- Easy to play with the code
  - Setup
  - Confident
- Easy to test
- Easy to change
- Functional core, imperative shell

Identity, state, value

— Atom for state

Why does it useful?

- Separate current vs facts
  - Easy to keep track changes overtime
  - Report
  - Undo
- Thread-safe

Macro

- Extend language
- when
- time
- doto

Why Macro?

- Small & stable core
- Build your own new feature without waiting for language owner
  - Goroutine & Channels as a library

Webserver

ClojureScript

- Google Closure Compiler and Libraries
  - Dead-code elimination
- With help of React.js
  - f(HTML) = DOM
- I think it's easier than React

ClojureScript

- Popular library are provided via cljsjs project
- Aiming for access to all NPM libraries
- Sometimes faster than React
  - Inspiration of Immutable.js

ClojureScript + React Native

Many mores!

- Property based testing (Generative testing)
- Sharing code between Cloure and ClojureScript and others
  - Same code for server & client rendering
  - Sync algorithm
- clojure.spec
- Datomic
  - Database as a value
  - Never forget

What to expect

- Can't program in the same way
  - Can't mutate
- Hard to read in the beginning
- Bad error message
- Slow start REPL
  - Don't need to restart often
- Learn host platform
- Less documentations
  - Do ask on FB group!

Conclusion

- Functional
- Lisp
- Data-oriented
- Reaches
- Again, FB: Clojure in Thai

Thank you

Questions?