ClojureScript

Moritz Ulrich

FrOSCon 2012

Outline

Hello World!

- Clojure

```
(println "Hello World!")
```

- ClojureScript

```
(.log js/console "Hello World!")
```

Hello World v2

- With `clojure.core/*print-fn*` bound to a function:

```
(println "Hello World!")
```

Clojure?

- Modern LISP
- Targets the JVM
  - Great Java-Interop
- Persistent Data Structures
- Easy concurrent programming
  - Built-in STM
  - Reactive Actors ('Agents')
  - Good encapsulation of State

ClojureScript?

- All advantages of Clojure
  - Functional Programming
  - Persistent Data Structures
  - Powerful concise Syntax
  - Great Host Interop
- Runs in the Browser
  - Wide target range
- No `eval`
  - Reader still available!

ClojureScript!

- Created by Rich Hickey in 2011
- Clojure Compiler which targets Javascript
- Same Runtime as Clojure
  - Data Structures
  - clojure.core
  - Code Sharing possible
- Cleaner Codebase
  - 100% Clojure, no Java/Javascript
  - Much of Clojure is implemented in Java

Proof?

```
find .  -name *.js
```

```
./samples/hello-js/externed-lib.js
./samples/hello-js/externs.js
./samples/hello-js/my-external-lib.js
./src/cljs/cljs/nodejs_externs.js
```

```
cat ./src/cljs/cljs/nodejs_externs.js
```

```
function require(){}
function process(){}
```

Why target Javascript?

- Faster startup time
- Widespread platform (Browsers)
- Node.js
- Advantages when using same code Server- and Client-Side

Platform Interop

- Function calls

```
(.log js/console "Foobar")
```

```
(js/alert 42)
```

- Properties

```
(.-location js/window)
```

```
(set! (.-prop obj) "Foo")
```

Interop: Syntactic Sugar

- Double Dot

```
(.. ($ "#my-table")
    (children "tr")
    (children "td")
    (hide))
```

- Anonymous Functions

```
(fn [a b] (+ a b))
```

```
#(+ %1 %2)
```

DOM Manipulation

- Google Closure
- jQuery

Google Closure

Javascript Library by Google

- Advantages
  - Used by ClojureScript itself
  - Very rich library containing many kinds of UI elements
  - Integrates nicely with the Google Closure compiler
- Disadvantages
  - Hard to use
  - Usually hard to integrate in legacy codebases
  - As of May 2012: No way to set data-attributes

jQuery

- Advantages
  - More concise to use
  - Widely known
  - Nice wrappers available (jayq)
- Disadvantages
  - Syntax doesn't integrate very good
  - Uses own Array type which doesn't work out-of-the-box with Clojure's Sequence abstractions

Compilation

1. Reader
2. Macros
3. Analysis
4. Emission
5. Closure Compiler

Closure Compiler

- Optimizing compiler for Javascript
- Performs the following:
  - Warnings
  - Dead-code elimination
  - Optimization
- Code must be written in a very strict style
  - ClojureScript generates such code

Size of generated Code

Code

```
(ns foo.bar)

(defn ^:export greet [name]
  (js/alert (str "Hello, " name "!")))
```

Result

```
91914  out-advanced.js
724380 out-pretty.js
```

Limitations

- No Multithreading
  - Atoms and Refs are still useful
- Complicated use of macros
- Many Clojure libraries don't work without modifications

Available Libraries

- User-Interface
  - Google Closure
  - jQuery UI
- DOM Manipulation/Generation
  - jayq (jQuery)
  - crate/hiccups (Hiccup)
  - enfocus (Enlive)
- All other Javascript libraries

IDEs

- Editor Support
  - clojurescript-mode (Emacs)
- Build Tools
  - lein-cljsbuild (Leiningen)
  - cljs-watch
- REPL
  - `lein cljsbuild repl-{listen,rhino}`

ClojureScript One

- Sample single-page application
- Browser connected REPL
- Well documented
- Great (but complex) starting point

Future Plans

- Pluggable Backends
- Source Maps
- Reactive Programming

Recent: Pluggable Backends

Summer of Code Project by Raphael Amiard

- Lexer extracted from monolithic Compiler
- Compiler implemented as modular Backend
- Soon:
  - Lua
  - Python
  - C
  - Malbolge?

Source Maps

- Map from compiled Javascript to ClojureScript
- Great for debugging errors
- Implemented in Chrome, support in ClojureScript coming

Reactive Programming

- Most stuff happens in the DOM
- Manual DOM Manipulation is cumbersome
- Solution: Bind values of elements/data-structures to modified values of other data structures

Conclusions

Good for

- Single Page applications with much logic
- Re-use of code written for the server

Not so good for

- Small utility scripts (high file size)
- High performance code

Starting Points

- Clojure Google Group

  http://groups.google.com/group/clojure
- ClojureScript on Github

  https://github.com/clojure/clojurescript
  (Don't follow the 'Quick Start' Guide! Use lein-cljsbuild for building projects or starting a REPL.)
- lein-cljsbuild

  https://github.com/emezeske/lein-cljsbuild
- ClojureScript One

  http://clojurescriptone.com/

Links

Google Closure	https://developers.google.com/closure/
jQuery	http://jquery.com/
jayq	https://github.com/ibdknox/jayq
crate	https://github.com/ibdknox/crate
hiccups	https://github.com/teropa/hiccups
enfocus	https://github.com/ckirkendall/enfocus
cljs-watch	https://github.com/ibdknox/cljs-watch

So long. . .

. . . and thanks for all the fish

Thank you!

Contact

Moritz Ulrich

moritz@tarn-vedra.de
https://github.com/the-kenny/
http://twitter.com/the$_k$*enny*