ClojureScript-in-ClojureSc

localhost:8080/index.html#/about-me

About Me

Principal Software Engineer at LonoCloud

Probably best known for creating noVNC and websockify

Connect:

- @bus_kanaka
- @noVNC (noVNC & HTML5 related)
- github.com/kanaka
- blog.n01se.net/kanaka
- cljs@martintribe.org

ClojureScript-in-ClojureSc ×

localhost:8080/index.html#/intro

Introducing ClojureScript-in-ClojureScript

A (hopefully temporary) fork of ClojureScript: https://github.com/kanaka/clojurescript

A Port of ClojureScript to ClojureScript

More specifically: A port of the Clojure parts of the ClojureScript compiler to ClojureScript.

ClojureScript-in-ClojureSc ×

localhost:8080/index.html#/why

Why?

The best reason I can give is to show examples of what is possible

Web Examples

Node.js Examples

ClojureScript-in-ClojureSc ×

localhost:8080/index.html#/why-web

Web Examples

- Online web REPL: ClojureScript.net

ClojureScript-in-ClojureSc × ClojureScript-in-ClojureSc ×

clojurescript.net

CLOJURESCRIPT.NET

ClojureScript Web REPL

ClojureScript-in-ClojureScript Web REPL

```
cljs.user=>
```

Show file editor

View source on Github

ClojureScript at a glance - PDF | Searchable docs | Translations from JavaScript

Datatypes

MAP	{:key1 :val1, :key2 :val2}
VECTORS	[1 2 3 4 :a :b :c 1 2]
SETS	#{:a :b :c 1 2 3}
SCALARS	a-symbol, :a-keyword, "a string"

Useful Functions

MATH	+ - * / quot rem mod inc dec max min
COMPARISON	= == not= < > <= >=
PREDICATES	nil? identical? zero? pos? neg? even? odd? true? false?

ClojureScript-in-ClojureSc × ClojureScript-in-ClojureSc ×

clojurescript.net

CLOJURESCRIPT.NET

ClojureScript Web REPL

ClojureScript-in-ClojureScript Web REPL

```
cljs.user=> (+ 1 1)
2
cljs.user=>
```

Show file editor

View source on Github

ClojureScript at a glance - PDF | Searchable docs | Translations from JavaScript

Datatypes

MAP	{:key1 :val1, :key2 :val2}
VECTORS	[1 2 3 4 :a :b :c 1 2]
SETS	#{:a :b :c 1 2 3}
SCALARS	a-symbol, :a-keyword, "a string"

Useful Functions

MATH	+ - * / quot rem mod inc dec max min
COMPARISON	= == not= < > <= >=
PREDICATES	nil? identical? zero? pos? neg? even? odd? true? false?

ClojureScript-in-ClojureSc × ClojureScript-in-ClojureSc ×

clojurescript.net

CLOJURESCRIPT.NET

ClojureScript Web REPL

ClojureScript-in-ClojureScript Web REPL

```
cljs.user=> (+ 1 1)
2
cljs.user=> (defn frob [x] (apply + 1 (vals (meta x))))
#<function frob(x){
return cljs.core.apply.call(null,cljs.core._PLUS_,1,cljs.core.vals.call(null,cljs.core.meta.call(null,x)));
}>
cljs.user=>
```

Show file editor

View source on Github

ClojureScript at a glance - PDF | Searchable docs | Translations from JavaScript

Datatypes

MAP	{:key1 :val1, :key2 :val2}
VECTORS	[1 2 3 4 :a :b :c 1 2]
SETS	#{:a :b :c 1 2 3}
SCALARS	a-symbol, :a-keyword, "a string"

Useful Functions

MATH	+ - * / quot rem mod inc dec max min
COMPARISON	= == not= < > <= >=
PREDICATES	nil? identical? zero? pos? neg? even? odd? true? false?

ClojureScript-in-ClojureSc × ClojureScript-in-ClojureSc ×

clojurescript.net

CLOJURESCRIPT.NET

ClojureScript Web REPL

ClojureScript-in-ClojureScript Web REPL

```
cljs.user=> (+ 1 1)
2
cljs.user=> (defn frob [x] (apply + 1 (vals (meta x))))
#<function frob(x){
return cljs.core.apply.call(null,cljs.core._PLUS_,1,cljs.core.vals.call(null,cljs.core.meta.call(null,x)));
}>
cljs.user=> (frob (with-meta [1 2 3] {:a 10 :b 20 :c 30}))
61
cljs.user=>
```

Show file editor

View source on Github

ClojureScript at a glance - PDF | Searchable docs | Translations from JavaScript

Datatypes

MAP	{:key1 :val1, :key2 :val2}
VECTORS	[1 2 3 4 :a :b :c 1 2]
SETS	#{:a :b :c 1 2 3}
SCALARS	a-symbol, :a-keyword, "a string"

Useful Functions

MATH	+ - * / quot rem mod inc dec max min
COMPARISON	= == not= < > <= >=
PREDICATES	nil? identical? zero? pos? neg? even? odd? true? false?

ClojureScript-in-ClojureSc × ClojureScript-in-ClojureSc ×

clojurescript.net

CLOJURESCRIPT.NET

ClojureScript Web REPL

ClojureScript-in-ClojureScript Web REPL

```
cljs.user=> (+ 1 1)
2
cljs.user=> (defn frob [x] (apply + 1 (vals (meta x))))
#<function frob(x){
return cljs.core.apply.call(null,cljs.core._PLUS_,1,cljs.core.vals.call(null,cljs.core.meta.call(null,x)));
}>
cljs.user=> (frob (with-meta [1 2 3] {:a 10 :b 20 :c 30}))
61
cljs.user=> (defmacro surprise-ending [s f] `(lazy-cat ~s (do ~f nil)))
nil
cljs.user=>
```

Show file editor

View source on Github

ClojureScript at a glance - PDF | Searchable docs | Translations from JavaScript

Datatypes

MAP	{:key1 :val1, :key2 :val2}
VECTORS	[1 2 3 4 :a :b :c 1 2]
SETS	#{:a :b :c 1 2 3}
SCALARS	a-symbol, :a-keyword, "a string"

Useful Functions

MATH	+ - * / quot rem mod inc dec max min
COMPARISON	= == not= < > <= >=
PREDICATES	nil? identical? zero? pos? neg? even? odd? true? false?

CLOJURESCRIPT.NET

ClojureScript Web REPL

ClojureScript-in-ClojureScript Web REPL

```
cljs.user=> (+ 1 1)
2
cljs.user=> (defn frob [x] (apply + 1 (vals (meta x))))
#<function frob(x){
return cljs.core.apply.call(null,cljs.core._PLUS_,1,cljs.core.vals.call(null,cljs.core.meta.call(null,x)));
}>
cljs.user=> (frob (with-meta [1 2 3] {:a 10 :b 20 :c 30}))
61
cljs.user=> (defmacro surprise-ending [s f] `(lazy-cat ~s (do ~f nil)))
nil
cljs.user=> (def things (surprise-ending [1 2 3] (println "SURPRISE!"))) nil
nil
cljs.user=>
```

Show file editor

View source on Github

ClojureScript at a glance - PDF | Searchable docs | Translations from JavaScript

Datatypes

MAP	{:key1 :val1, :key2 :val2}
VECTORS	[1 2 3 4 :a :b :c 1 2]
SETS	#{:a :b :c 1 2 3}
SCALARS	a-symbol, :a-keyword, "a string"

Useful Functions

MATH	+ - * / quot rem mod inc dec max min
COMPARISON	= == not= < > <= >=
PREDICATES	nil? identical? zero? pos? neg? even? odd? true? false?

CLOJURESCRIPT.NET

ClojureScript Web REPL

```
cljs.user=> (+ 1 1)
2
cljs.user=> (defn frob [x] (apply + 1 (vals (meta x))))
#<function frob(x){
return cljs.core.apply.call(null,cljs.core._PLUS_,1,cljs.core.vals.call(null,cljs.core.meta.call(null,x)));
}>
cljs.user=> (frob (with-meta [1 2 3] {:a 10 :b 20 :c 30}))
61
cljs.user=> (defmacro surprise-ending [s f] `(lazy-cat ~s (do ~f nil)))
nil
cljs.user=> (def things (surprise-ending [1 2 3] (println "SURPRISE!"))) nil
nil
cljs.user=> (take 2 things)
(1 2)
cljs.user=>
```

Show file editor

View source on Github

ClojureScript at a glance - PDF | Searchable docs | Translations from JavaScript

Datatypes

MAP	{:key1 :val1, :key2 :val2}
VECTORS	[1 2 3 4 :a :b :c 1 2]
SETS	#{:a :b :c 1 2 3}
SCALARS	a-symbol, :a-keyword, "a string"

Useful Functions

MATH	+ - * / quot rem mod inc dec max min
COMPARISON	= == not= < > <= >=
PREDICATES	nil? identical? zero? pos? neg? even? odd? true? false?

CLOJURESCRIPT.NET

ClojureScript Web REPL

```
cljs.user=> (defn frob [x] (apply + 1 (vals (meta x))))
#<function frob(x){
return cljs.core.apply.call(null,cljs.core._PLUS_,1,cljs.core.vals.call(null,cljs.core.meta.call(null,x)));
}>
cljs.user=> (frob (with-meta [1 2 3] {:a 10 :b 20 :c 30}))
61
cljs.user=> (defmacro surprise-ending [s f] `(lazy-cat ~s (do ~f nil)))
nil
cljs.user=> (def things (surprise-ending [1 2 3] (println "SURPRISE!"))) nil
nil
cljs.user=> (take 2 things)
(1 2)
cljs.user=> (take 3 things)
(1 2 3)
cljs.user=>
```

Show file editor

View source on Github

ClojureScript at a glance - PDF | Searchable docs | Translations from JavaScript

Datatypes

MAP	{:key1 :val1, :key2 :val2}
VECTORS	[1 2 3 4 :a :b :c 1 2]
SETS	#{:a :b :c 1 2 3}
SCALARS	a-symbol, :a-keyword, "a string"

Useful Functions

MATH	+ - * / quot rem mod inc dec max min
COMPARISON	= == not= < > <= >=
PREDICATES	nil? identical? zero? pos? neg? even? odd? true? false?

ClojureScript-in-ClojureSc × ClojureScript-in-ClojureSc ×

clojurescript.net

CLOJURESCRIPT.NET

ClojureScript Web REPL

```
}>
cljs.user=> (frob (with-meta [1 2 3] {:a 10 :b 20 :c 30}))
61
cljs.user=> (defmacro surprise-ending [s f] `(lazy-cat ~s (do ~f nil)))
nil
cljs.user=> (def things (surprise-ending [1 2 3] (println "SURPRISE!"))) nil
nil
cljs.user=> (take 2 things)
(1 2)
cljs.user=> (take 3 things)
(1 2 3)
cljs.user=> (take 4 things)
SURPRISE!
(1 2 3)
cljs.user=>
```

Show file editor

View source on Github

ClojureScript at a glance - PDF | Searchable docs | Translations from JavaScript

Datatypes

MAP	{:key1 :val1, :key2 :val2}
VECTORS	[1 2 3 4 :a :b :c 1 2]
SETS	#{:a :b :c 1 2 3}
SCALARS	a-symbol, :a-keyword, "a string"

Useful Functions

MATH	+ - * / quot rem mod inc dec max min
COMPARISON	= == not= < > <= >=
PREDICATES	nil? identical? zero? pos? neg? even? odd? true? false?

CLOJURESCRIPT.NET

ClojureScript Web REPL

```
61
cljs.user=> (defmacro surprise-ending [s f] `(lazy-cat ~s (do ~f nil)))
nil
cljs.user=> (def things (surprise-ending [1 2 3] (println "SURPRISE!"))) nil
nil
cljs.user=> (take 2 things)
(1 2)
cljs.user=> (take 3 things)
(1 2 3)
cljs.user=> (take 4 things)
SURPRISE!
(1 2 3)
cljs.user=> (take 4 things)
(1 2 3)
cljs.user=>
```

Show file editor

View source on Github

ClojureScript at a glance - PDF | Searchable docs | Translations from JavaScript

Datatypes

MAP	{:key1 :val1, :key2 :val2}
VECTORS	[1 2 3 4 :a :b :c 1 2]
SETS	#{:a :b :c 1 2 3}
SCALARS	a-symbol, :a-keyword, "a string"

Useful Functions

MATH	+ - * / quot rem mod inc dec max min
COMPARISON	= == not= < > <= >=
PREDICATES	nil? identical? zero? pos? neg? even? odd? true? false?

CLOJURESCRIPT.NET

ClojureScript Web REPL

```
nil
cljs.user=> (def things (surprise-ending [1 2 3] (println "SURPRISE!"))) nil
nil
cljs.user=> (take 2 things)
(1 2)
cljs.user=> (take 3 things)
(1 2 3)
cljs.user=> (take 4 things)
SURPRISE!
(1 2 3)
cljs.user=> (take 4 things)
(1 2 3)
cljs.user=> (take 4 things)
(1 2 3)
cljs.user=>
```

Show file editor

View source on Github

ClojureScript at a glance - PDF | Searchable docs | Translations from JavaScript

Datatypes

MAP	{:key1 :val1, :key2 :val2}
VECTORS	[1 2 3 4 :a :b :c 1 2]
SETS	#{:a :b :c 1 2 3}
SCALARS	a-symbol, :a-keyword, "a string"

Useful Functions

MATH	+ - * / quot rem mod inc dec max min
COMPARISON	= == not= < > <= >=
PREDICATES	nil? identical? zero? pos? neg? even? odd? true? false?

Web Examples

- Online web REPL: ClojureScript.net

```
(+ 1 1)
(defn frob [x] (apply + 1 (vals (meta x))))
(frob (with-meta [1 2 3] {:a 10 :b 20 :c 30}))
(defmacro surprise-ending [s f]
    `(lazy-cat ~s (do ~f nil)))
(def things (surprise-ending [1 2 3]
    (println "SURPRISE!"))) nil
```

  - NOTE: trailing nil is necessary

```
(take 2 things)
(take 3 things)
(take 4 things)
(take 4 things)
```

Web Examples

- Online web REPL: ClojureScript.net

```
(+ 1 1)
(defn frob [x] (apply + 1 (vals (meta x))))
(frob (with-meta [1 2 3] {:a 10 :b 20 :c 30}))
(defmacro surprise-ending [s f]
    `(lazy-cat ~s (do ~f nil)))
(def things (surprise-ending [1 2 3]
    (println "SURPRISE!"))) nil
```

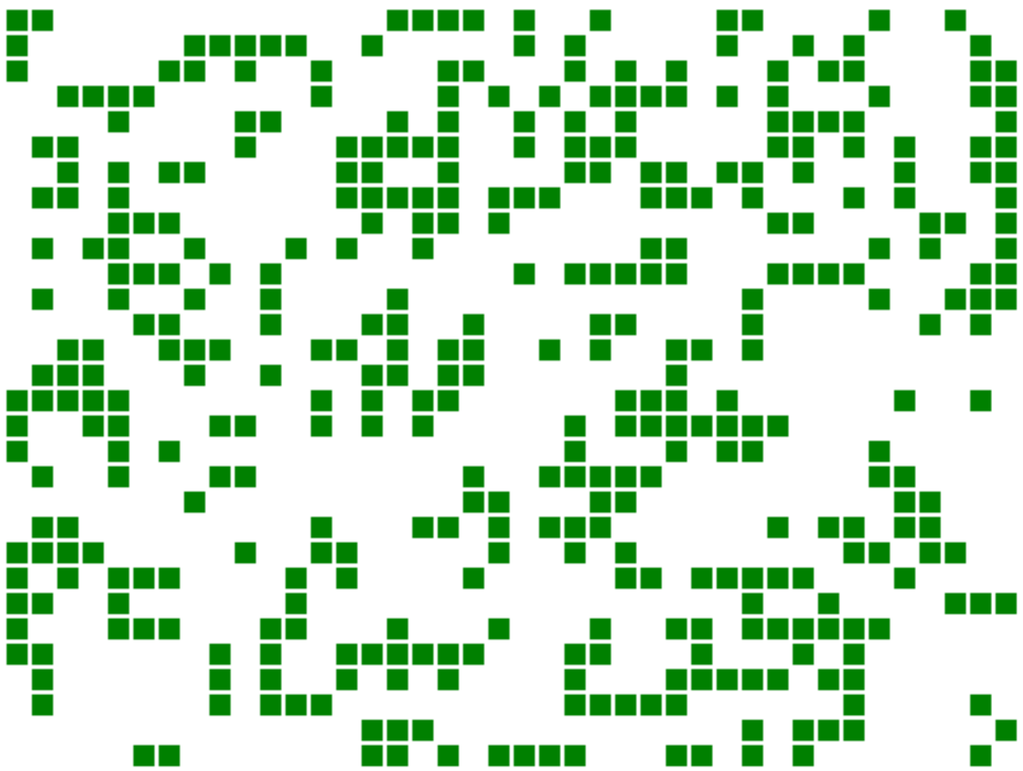
  - NOTE: trailing nil is necessary

```
(take 2 things)
(take 3 things)
(take 4 things)
(take 4 things)
```

- Debug Web REPL for ClojureScript Projects: ClojureScript Game of Life

ClojureScript-in-ClojureSc × | ClojureScript-in-ClojureSc × | localhost:8080/game-of-li ×

localhost:8080/game-of-life.html



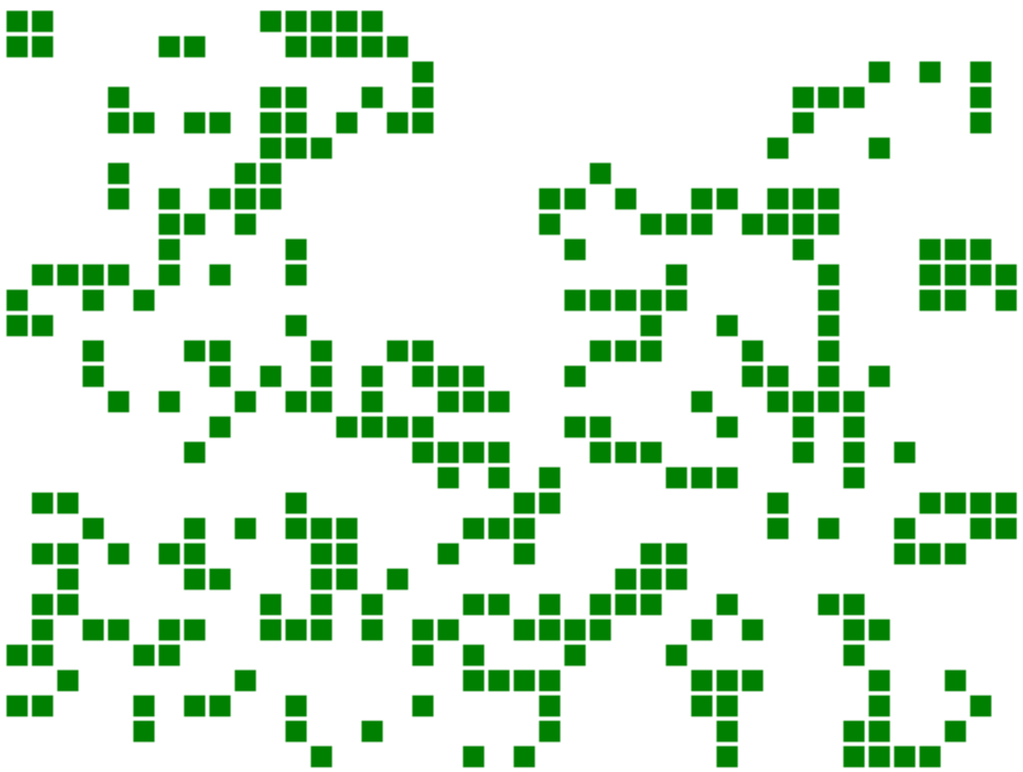
Copy/paste at REPL:

```
@ldnclj_dojo_team_5_game_
(count @ldnclj_dojo_team_5_
(reset! ldnclj_dojo_team_5_g
```

DEBUG REPL

ClojureScript-in-ClojureSc | ClojureScript-in-ClojureSc | localhost:8080/game-of-li

localhost:8080/game-of-life.html



ClojureScript-in-ClojureScript Web REPL
cljs.user=>

Copy/paste at REPL:

@ldnclj_dojo_team_5_game_

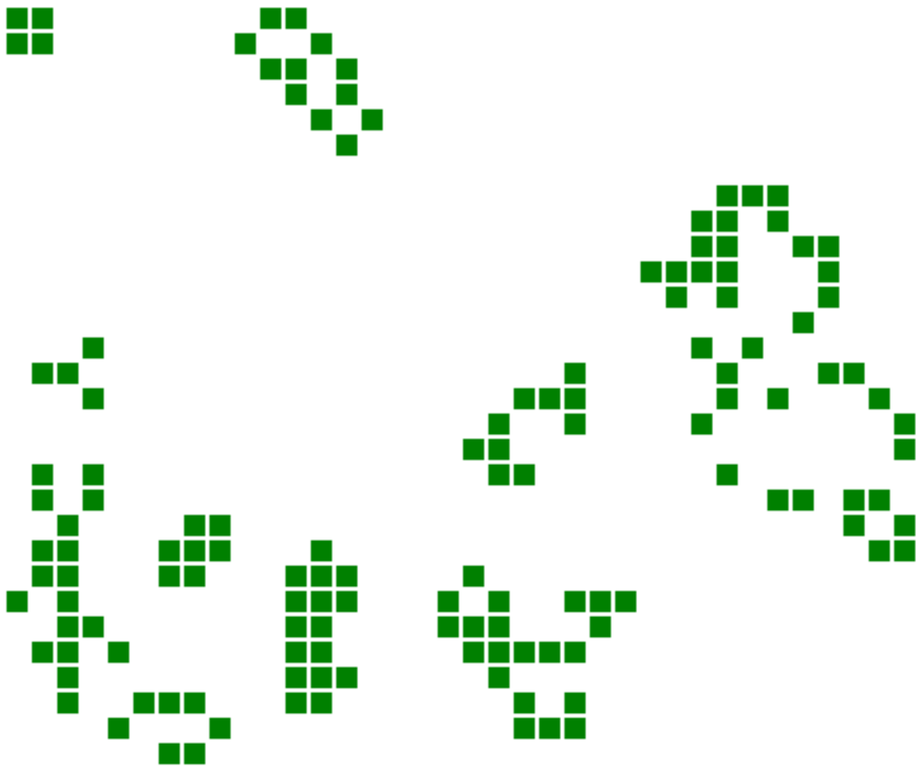
(count @ldnclj_dojo_team_5_

(reset! ldnclj_dojo_team_5_g

DEBUG REPL

ClojureScript-in-ClojureSc | ClojureScript-in-ClojureSc | localhost:8080/game-of-li

localhost:8080/game-of-life.html



ClojureScript-in-ClojureScript Web REPL
cljs.user=> @ldnclj_dojo_team_5_game_of_life.model/*grid*

Copy/paste at REPL:

@ldnclj_dojo_team_5_game_

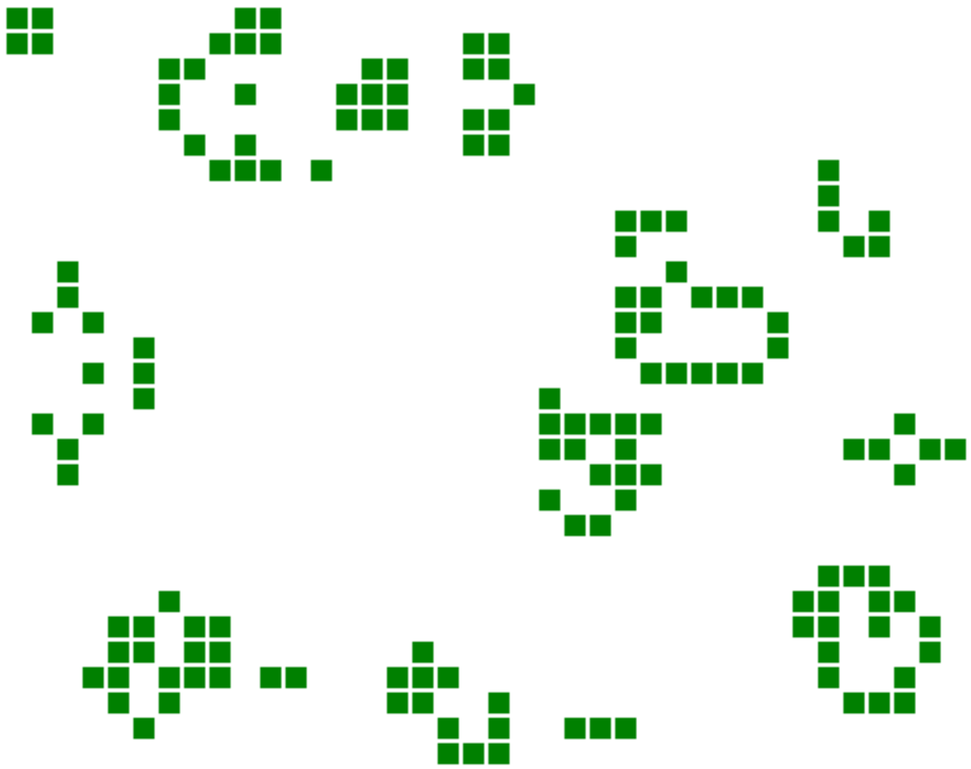
(count @ldnclj_dojo_team_5_

(reset! ldnclj_dojo_team_5_g

DEBUG REPL

ClojureScript-in-ClojureSc × | ClojureScript-in-ClojureSc × | localhost:8080/game-of-li ×

localhost:8080/game-of-life.html



Copy/paste at REPL:

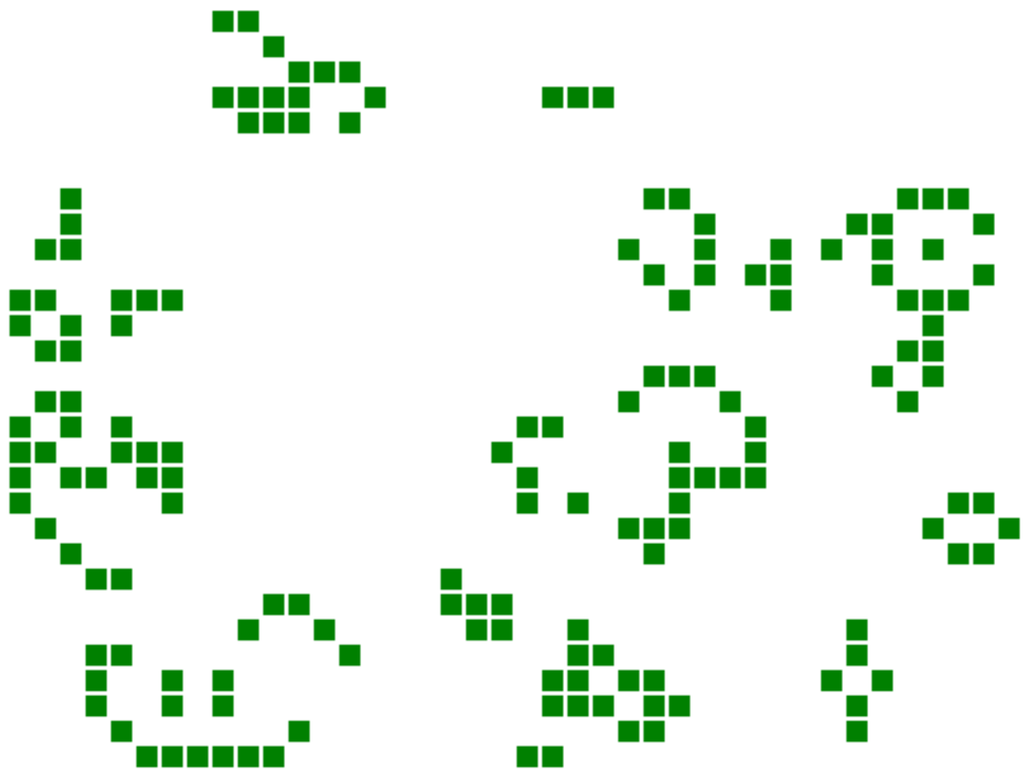
@ldnclj_dojo_team_5_game_

(count @ldnclj_dojo_team_5_

(reset! ldnclj_dojo_team_5_g

```
nil nil :alive :alive :alive nil nil nil nil nil nil nil nil nil nil :alive
:alive :alive nil nil nil nil nil nil nil nil nil nil :alive :alive nil :alive
nil nil nil nil nil nil nil :alive nil nil nil nil nil nil nil nil nil nil nil
nil nil :alive nil nil nil nil nil nil nil nil nil nil nil :alive :alive
nil :alive :alive :alive :alive nil nil :alive nil nil nil :alive nil nil
nil nil nil nil nil nil :alive nil :alive :alive nil nil nil nil nil nil nil
nil nil nil :alive nil :alive nil nil nil :alive nil nil :alive :alive nil :alive
nil nil nil nil nil nil nil nil nil nil nil nil :alive :alive :alive nil nil
nil nil nil nil nil :alive :alive nil :alive nil nil :alive nil nil nil nil nil nil
:alive nil nil :alive nil nil nil :alive nil nil nil nil nil nil nil nil nil :alive
nil nil nil nil nil nil nil nil nil nil nil :alive nil :alive :alive nil nil
nil :alive nil nil :alive nil nil nil nil nil nil :alive nil nil nil nil nil nil
nil nil nil nil nil nil nil nil nil nil nil nil nil :alive nil nil nil nil nil :alive
:alive nil nil nil nil :alive :alive nil nil nil nil :alive nil nil :alive nil nil
nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil :alive nil nil nil
nil :alive :alive :alive nil nil nil nil nil nil nil nil nil :alive :alive nil
:alive nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil
nil nil nil nil :alive :alive nil nil nil nil nil nil nil nil nil nil :alive
nil nil :alive nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil
nil nil nil :alive nil nil :alive nil nil nil nil nil nil nil nil nil nil nil
nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil]
cljs.user=>
```

DEBUG REPL

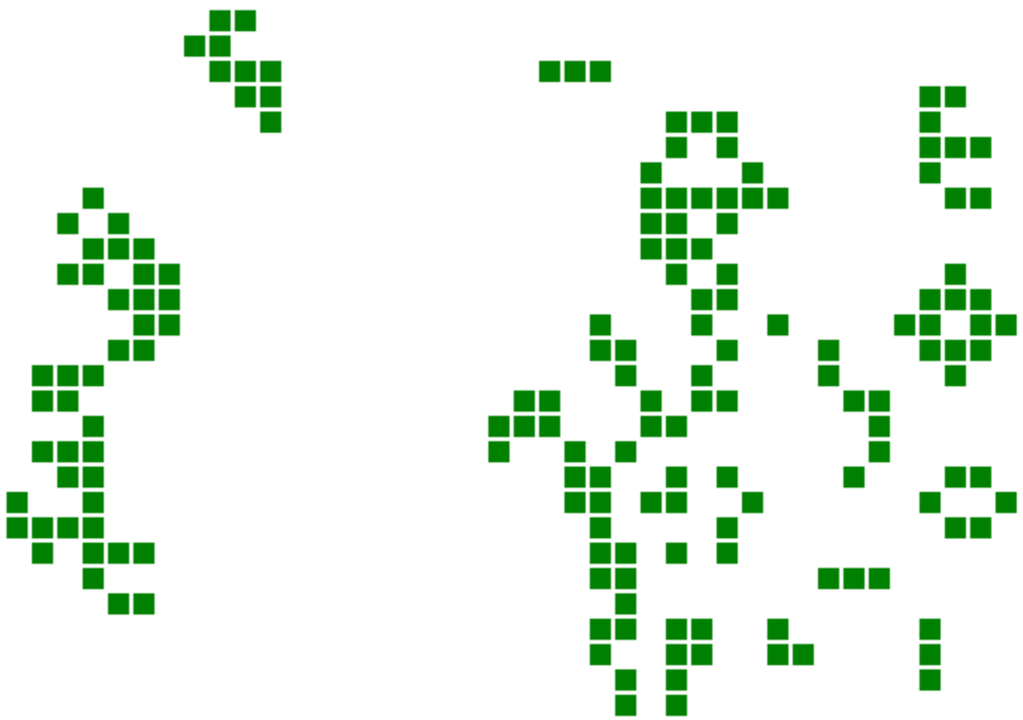


```
nil nil :alive :alive :alive nil nil nil nil nil nil nil nil nil nil :alive
:alive :alive nil nil nil nil nil nil nil nil nil nil :alive :alive nil :alive
nil nil nil nil nil nil nil :alive nil nil nil nil nil nil nil nil nil nil nil
nil nil :alive nil nil nil nil nil nil nil nil nil nil nil :alive :alive
nil :alive :alive :alive :alive nil nil :alive nil nil nil :alive nil nil
nil nil nil nil nil nil :alive nil :alive :alive nil nil nil nil nil nil nil
nil nil nil :alive nil :alive nil nil nil :alive nil nil :alive :alive nil :alive
nil nil nil nil nil nil nil nil nil nil nil :alive :alive :alive nil nil nil
nil nil nil nil nil :alive :alive nil :alive nil nil :alive nil nil nil nil nil nil
:alive nil nil :alive nil nil nil :alive nil nil nil nil nil nil nil nil :alive
nil nil nil nil nil nil nil nil nil nil nil :alive nil :alive :alive nil nil
nil :alive nil nil :alive nil nil nil nil nil nil :alive nil nil nil nil nil nil
nil nil nil nil nil nil nil nil nil nil nil :alive nil nil nil nil nil :alive
:alive nil nil nil nil :alive :alive nil nil nil nil :alive nil nil :alive nil nil
nil nil nil nil nil nil nil nil nil nil nil nil nil nil :alive nil nil nil
nil :alive :alive :alive nil nil nil nil nil nil nil nil :alive :alive nil
:alive nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil
nil nil nil nil :alive :alive nil nil nil nil nil nil nil nil nil nil :alive
nil nil :alive nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil
nil nil nil :alive nil nil :alive nil nil nil nil nil nil nil nil nil nil nil
nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil]
cljs.user=> (count @ldnclj_dojo_team_5_game_of_life.model/*grid*)
```

Copy/paste at REPL:

```
@ldnclj_dojo_team_5_game_
(count @ldnclj_dojo_team_5_
(reset! ldnclj_dojo_team_5_g
```

DEBUG REPL



```
nil nil nil nil nil nil nil :alive nil nil nil nil nil nil nil nil nil nil nil
nil nil :alive nil nil nil nil nil nil nil nil nil nil nil nil :alive :alive
nil :alive :alive :alive :alive nil nil :alive nil nil nil :alive nil nil
nil nil nil nil nil nil :alive nil :alive :alive nil nil nil nil nil nil nil
nil nil nil :alive nil :alive nil nil nil :alive nil nil :alive :alive nil :alive
nil nil nil nil nil nil nil nil nil nil nil :alive :alive :alive nil nil nil
nil nil nil nil nil :alive :alive nil :alive nil nil :alive nil nil nil nil nil nil
:alive nil nil :alive nil nil nil :alive nil nil nil nil nil nil nil nil :alive
nil nil nil nil nil nil nil nil nil nil nil :alive nil :alive :alive nil nil
nil :alive nil nil :alive nil nil nil nil nil nil :alive nil nil nil nil nil nil
nil nil nil nil nil nil nil nil nil nil nil nil nil nil :alive nil nil nil nil :alive
:alive nil nil nil nil :alive :alive nil nil nil nil :alive nil nil :alive nil nil
nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil :alive nil nil nil
nil :alive :alive :alive nil nil nil nil nil nil nil nil nil :alive :alive nil
:alive nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil
nil nil nil nil :alive :alive nil nil nil nil nil nil nil nil nil nil :alive
nil nil :alive nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil
nil nil nil :alive nil nil :alive nil nil nil nil nil nil nil nil nil nil nil nil
nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil]
cljs.user=> (count @ldnclj_dojo_team_5_game_of_life.model/*grid*)
1200
cljs.user=>
```

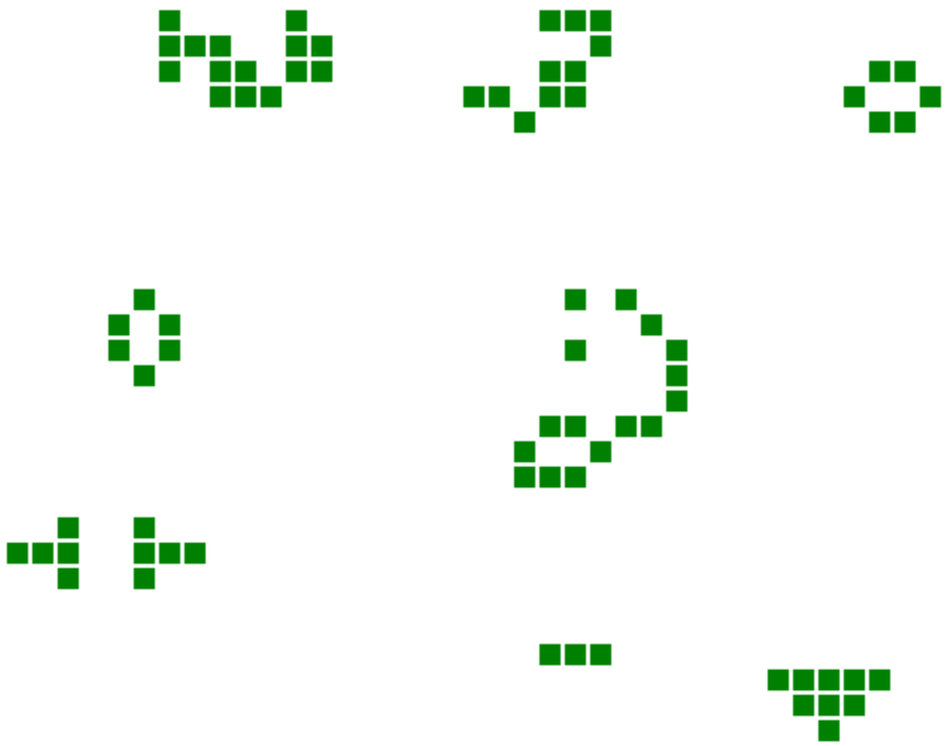
Copy/paste at REPL:

```
@ldnclj_dojo_team_5_game_
(count @ldnclj_dojo_team_5_
(reset! ldnclj_dojo_team_5_g
```

DEBUG REPL

ClojureScript-in-ClojureSc × | ClojureScript-in-ClojureSc × | localhost:8080/game-of-li ×

localhost:8080/game-of-life.html



```
nil nil :alive nil nil nil nil nil nil nil nil nil nil nil :alive :alive
nil :alive :alive :alive :alive :alive nil nil :alive nil nil nil :alive nil nil
nil nil nil nil nil nil :alive nil :alive :alive nil nil nil nil nil nil nil nil
nil nil nil :alive nil :alive nil nil nil :alive nil nil :alive :alive nil :alive
nil nil nil nil nil nil nil nil nil nil nil nil :alive :alive :alive nil nil nil
nil nil nil nil nil :alive :alive nil :alive nil nil :alive nil nil nil nil nil nil
:alive nil nil :alive nil nil nil :alive nil nil nil nil nil nil nil nil nil :alive
nil nil nil nil nil nil nil nil nil nil nil :alive nil :alive :alive nil nil nil
nil :alive nil nil :alive nil nil nil nil nil nil :alive nil nil nil nil nil nil nil
nil nil nil nil nil nil nil nil nil nil nil nil nil :alive nil nil nil nil nil :alive
:alive nil nil nil nil :alive :alive nil nil nil nil :alive nil nil :alive nil nil
nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil :alive nil nil nil
nil :alive :alive :alive nil nil nil nil nil nil nil nil nil :alive :alive nil
:alive nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil
nil nil nil nil :alive :alive nil nil nil nil nil nil nil nil nil nil :alive
nil nil :alive nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil
nil nil nil :alive nil nil :alive nil nil nil nil nil nil nil nil nil nil nil nil nil
nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil nil]
cljs.user=> (count @ldnclj_dojo_team_5_game_of_life.model/*grid*)
1200
cljs.user=> (reset! ldnclj_dojo_team_5_game_of_life.model/*grid* (vec (repeatedly
1200 #(rand-nth [nil nil :alive]))))
```

Copy/paste at REPL:

@ldnclj_dojo_team_5_game_

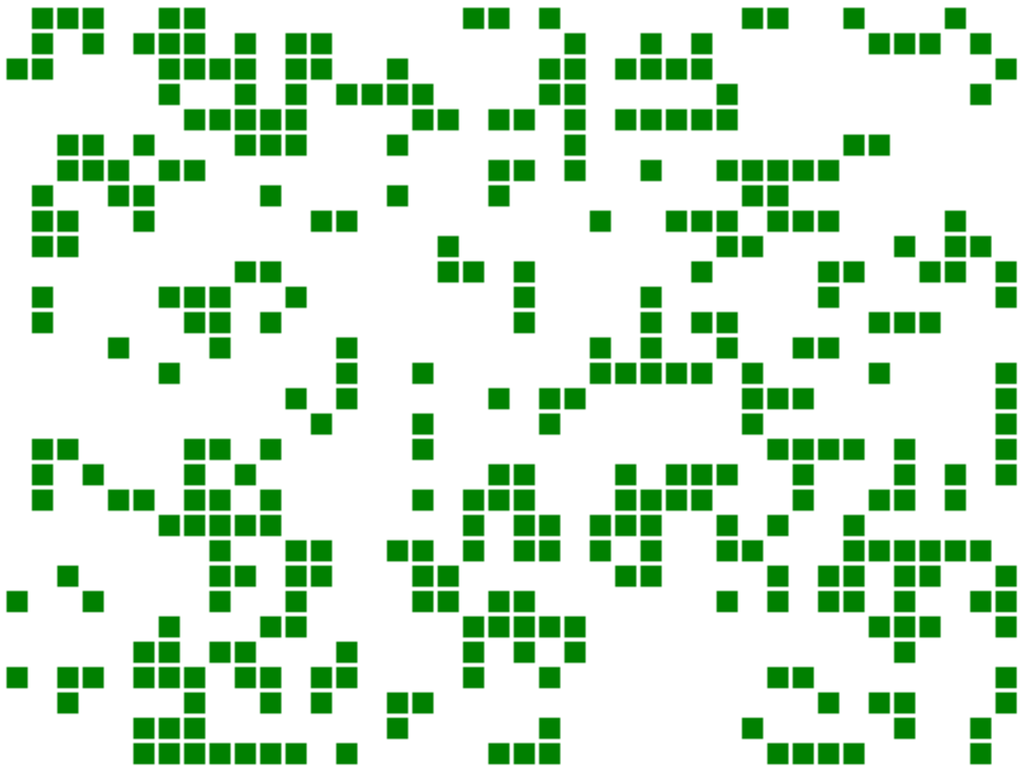
(count @ldnclj_dojo_team_5_

(reset! ldnclj_dojo_team_5_g

DEBUG REPL

ClojureScript-in-ClojureSc × | ClojureScript-in-ClojureSc × | localhost:8080/game-of-li ×

localhost:8080/game-of-life.html

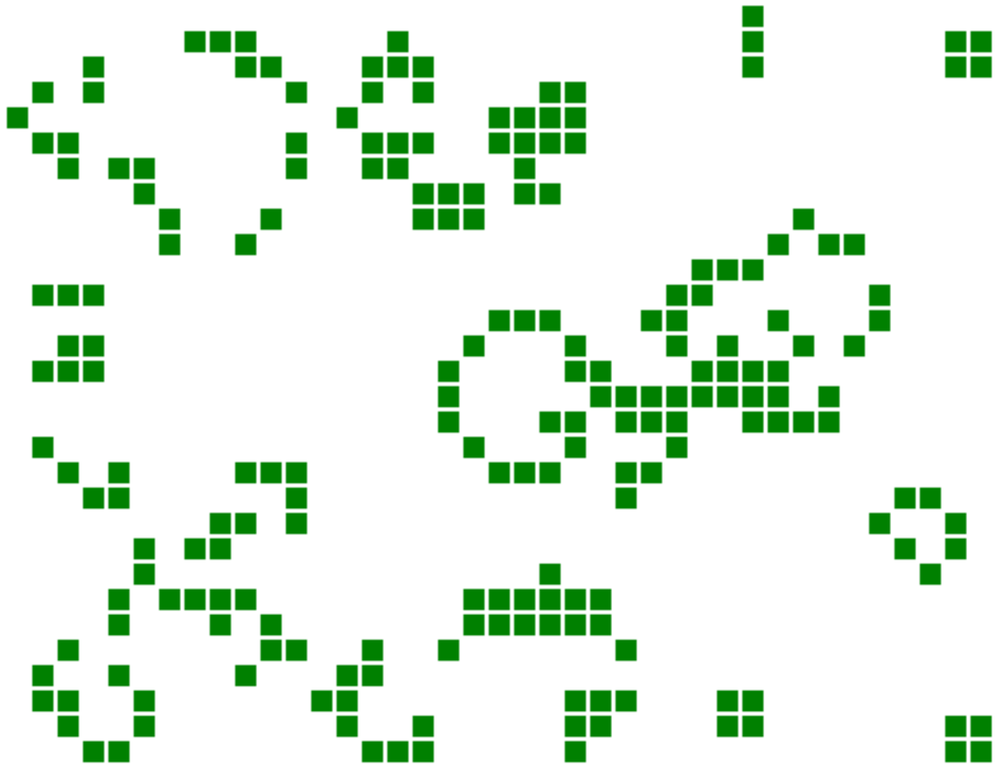


Copy/paste at REPL:

```
@ldnclj_dojo_team_5_game_
(count @ldnclj_dojo_team_5_
(reset! ldnclj_dojo_team_5_g
```

```
nil nil nil nil :alive :alive nil nil :alive nil nil :alive :alive nil :alive nil
nil nil nil nil nil nil nil :alive nil :alive nil nil :alive nil nil nil :alive
:alive nil :alive nil nil nil nil nil nil nil nil nil :alive nil :alive nil :alive
:alive nil :alive nil nil :alive :alive nil nil nil :alive nil nil :alive :alive
nil nil nil nil nil :alive :alive nil :alive nil :alive nil nil :alive :alive nil
nil :alive :alive nil nil nil :alive nil :alive :alive :alive :alive nil nil :alive
nil nil nil nil nil nil :alive nil :alive nil nil :alive nil nil nil nil nil nil
nil nil nil nil nil nil nil nil nil nil nil nil nil :alive nil nil :alive nil nil
:alive :alive nil nil nil :alive :alive :alive nil nil nil nil nil :alive nil
:alive nil nil nil nil :alive nil :alive :alive nil nil nil nil :alive :alive nil
nil nil nil nil nil nil nil nil :alive nil nil :alive nil :alive nil nil nil :alive
:alive nil nil nil nil nil nil nil :alive nil nil :alive nil nil nil :alive nil
:alive :alive :alive :alive :alive :alive nil :alive nil nil nil nil nil :alive
:alive nil :alive nil nil nil :alive nil :alive :alive :alive nil :alive nil nil
:alive nil nil nil nil :alive nil :alive nil nil nil nil :alive nil :alive :alive
nil nil nil nil :alive nil :alive nil nil nil nil nil nil :alive nil :alive nil nil
nil :alive nil :alive nil :alive nil nil :alive nil nil :alive :alive nil nil nil
nil nil :alive nil nil nil nil nil :alive nil nil nil :alive :alive :alive :alive
nil :alive nil nil :alive nil nil nil :alive :alive :alive nil :alive :alive nil
nil :alive nil nil nil :alive nil :alive :alive nil nil nil nil nil nil :alive
:alive :alive :alive nil nil nil nil nil nil]
cljs.user=>
```

DEBUG REPL



Copy/paste at REPL:

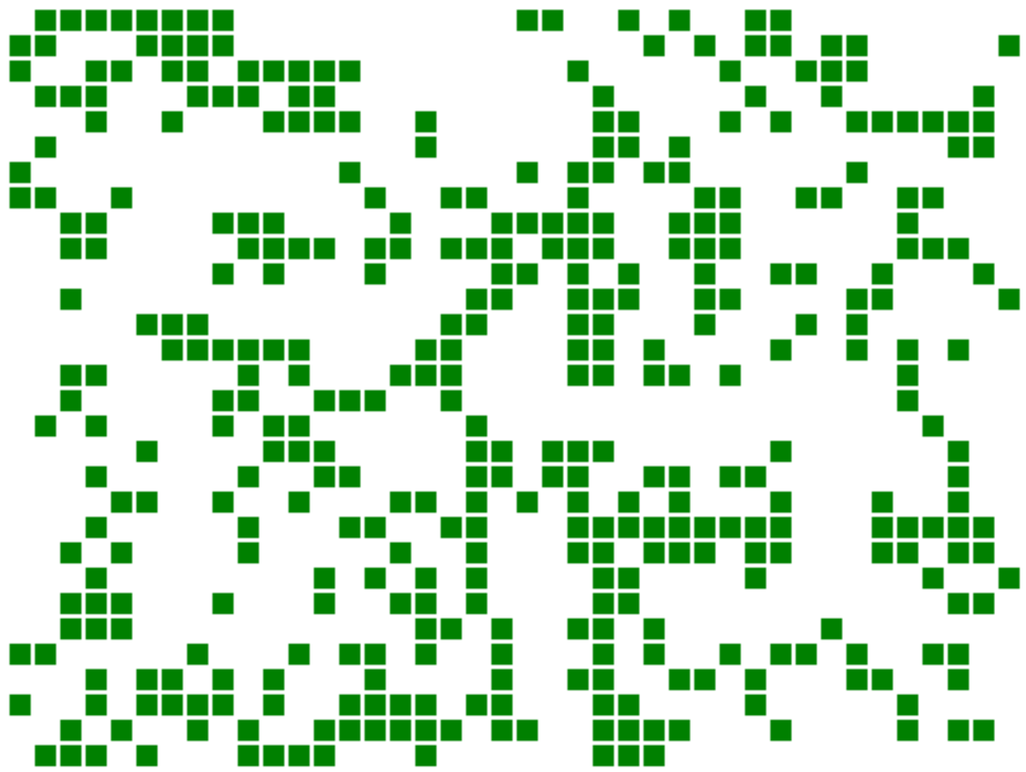
```
@ldnclj_dojo_team_5_game_
(count @ldnclj_dojo_team_5_
(reset! ldnclj_dojo_team_5_g
```

```
nil nil nil nil nil nil nil :alive nil :alive nil nil :alive nil nil nil :alive
:alive nil :alive nil nil nil nil nil nil nil nil :alive nil :alive nil :alive
:alive nil :alive nil nil :alive :alive nil nil nil :alive nil nil :alive :alive
nil nil nil nil nil :alive :alive nil :alive nil :alive nil nil :alive :alive nil
nil :alive :alive nil nil nil :alive nil :alive :alive :alive :alive nil nil :alive
nil nil nil nil nil nil :alive nil :alive nil nil :alive nil nil nil nil nil nil
nil nil nil nil nil nil nil nil nil nil nil nil :alive nil nil :alive nil nil
:alive :alive nil nil nil :alive :alive :alive nil nil nil nil nil :alive nil
:alive nil nil nil nil :alive nil :alive :alive nil nil nil nil :alive :alive nil
nil nil nil nil nil nil nil nil :alive nil nil :alive nil :alive nil nil nil :alive
:alive nil nil nil nil nil nil nil :alive nil nil :alive nil nil nil :alive nil
:alive :alive :alive :alive :alive :alive nil :alive nil nil nil nil nil :alive
:alive nil :alive nil nil nil :alive nil :alive :alive :alive nil :alive nil nil
:alive nil nil nil nil :alive nil :alive nil nil nil nil :alive nil :alive :alive
nil nil nil nil :alive nil :alive nil nil nil nil nil nil :alive nil :alive nil nil
nil :alive nil :alive nil :alive nil nil :alive nil nil :alive :alive nil nil nil
nil nil :alive nil nil nil nil nil :alive nil nil nil :alive :alive :alive :alive
nil :alive nil nil :alive nil nil nil :alive :alive :alive nil :alive :alive nil
nil :alive nil nil nil :alive nil :alive :alive nil nil nil nil nil nil :alive
:alive :alive :alive nil nil nil nil nil nil]
cljs.user=> (reset! ldnclj_dojo_team_5_game_of_life.model/*grid* (vec (repeatedly
1200 #(rand-nth [nil nil :alive]))))
```

DEBUG REPL

ClojureScript-in-ClojureSc × | ClojureScript-in-ClojureSc × | localhost:8080/game-of-li ×

localhost:8080/game-of-life.html



Copy/paste at REPL:

@ldnclj_dojo_team_5_game_

(count @ldnclj_dojo_team_5_

(reset! ldnclj_dojo_team_5_g

```
nil nil nil nil nil nil :alive :alive :alive :alive :alive nil :alive nil :alive
nil nil nil nil :alive nil nil :alive :alive :alive nil nil nil nil nil nil :alive
nil nil :alive nil nil :alive nil nil nil nil :alive nil nil :alive :alive nil nil
nil nil nil :alive :alive nil nil nil nil :alive :alive :alive nil nil nil nil nil
:alive nil nil :alive nil nil nil :alive nil nil nil nil nil nil nil nil nil nil
nil nil nil :alive nil nil nil nil nil nil nil nil :alive :alive :alive nil
:alive nil :alive nil nil :alive nil :alive nil nil nil :alive nil nil nil :alive
nil nil nil nil nil nil nil nil nil nil nil :alive :alive :alive nil :alive nil nil
nil :alive nil :alive nil nil nil nil nil nil nil :alive nil :alive nil :alive nil
:alive :alive :alive nil nil :alive nil nil :alive :alive :alive nil nil :alive nil
nil nil nil :alive :alive nil nil nil :alive nil :alive nil :alive nil nil :alive
nil :alive nil nil nil :alive nil nil nil :alive nil nil :alive :alive nil nil
:alive :alive nil :alive nil nil :alive :alive nil nil :alive :alive nil nil :alive
nil :alive :alive nil nil :alive nil nil nil :alive nil nil nil :alive :alive nil
nil nil nil nil nil :alive nil nil nil :alive :alive nil nil nil :alive :alive nil
nil nil nil :alive nil nil nil :alive nil nil :alive :alive nil nil :alive nil nil
:alive nil :alive nil nil nil nil nil nil :alive :alive nil :alive nil nil
:alive nil :alive nil nil nil nil :alive :alive nil :alive :alive :alive nil :alive
:alive :alive nil nil nil :alive :alive nil :alive nil nil :alive nil nil :alive
nil :alive nil nil :alive nil nil nil nil nil nil nil nil nil :alive nil nil
nil nil]
cljs.user=>
```

DEBUG REPL

ClojureScript-in-ClojureSc × ClojureScript-in-ClojureSc ×

localhost:8080/index.html#/why-web

Web Examples

- Online web REPL: ClojureScript.net

```
(+ 1 1)
(defn frob [x] (apply + 1 (vals (meta x))))
(frob (with-meta [1 2 3] {:a 10 :b 20 :c 30}))
(defmacro surprise-ending [s f]
    `(lazy-cat ~s (do ~f nil)))
(def things (surprise-ending [1 2 3]
    (println "SURPRISE!"))) nil
```

  - NOTE: trailing nil is necessary

```
(take 2 things)
(take 3 things)
(take 4 things)
(take 4 things)
```

- Debug Web REPL for ClojureScript Projects: ClojureScript Game of Life
- HTML Script tags with ClojureScript: Scriptany. Steps: 1, 2, 3, 4.

ClojureScript-in-ClojureSc | ClojureScript-in-ClojureSc | localhost:8080/scriptany-

localhost:8080/scriptany-cljs.html

Hello World!
Hint: view source

Elements | Resources | Network | Sources | Timeline | Profiles | Audits | Console

<top frame> | All | Errors | Warnings | Logs | Debug

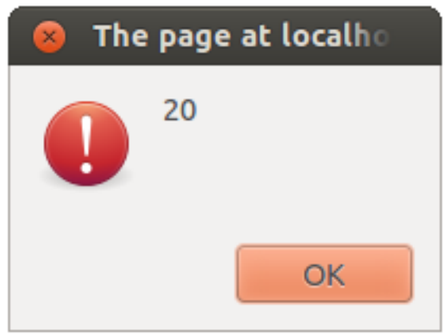
```
<!DOCTYPE html>
<html>
<head>
  <script type="text/javascript" src="scriptany.js"></script>
  <script src="repl-out/goog/base.js"></script>
  <script src="repl-out/repl.js"></script>
  <script>
      goog.require("cljs.repl");
  </script>
  <script>
      cljs.repl.init();
      scriptany.register("text/clojurescript", function (text)  {
          var ret = cljs.repl.read_eval_print(text);
          console.log("RTN:", ret);
      });
  </script>

</head>
<body>

    <div id="hello">Hello World!</div>

    <div>Hint: view source</div>

</body>
</html>
~
```

```
<!DOCTYPE html>
<html>
<head>
  <script type="text/javascript" src="scriptany.js"></script>
  <script src="repl-out/goog/base.js"></script>
  <script src="repl-out/repl.js"></script>
  <script>
      goog.require("cljs.repl");
  </script>
  <script>
      cljs.repl.init();
      scriptany.register("text/clojurescript", function (text)  {
          var ret = cljs.repl.read_eval_print(text);
          console.log("RTN:", ret);
      });
  </script>

  <script type="text/clojurescript">
      (defn foo [x] (* 2 x))
      (js/alert (foo 10))
  </script>

</head>
<body>

    <div id="hello">Hello World!</div>
```

ClojureScript-in-ClojureSc | ClojureScript-in-ClojureSc | localhost:8080/scriptany-

localhost:8080/scriptany-cljs.html

Hello World!
Hint: view source



Elements Resources Network Sources Timeline Profiles Audits Console

<top frame> All Errors Warnings Logs Debug

File Edit View Search Terminal Help

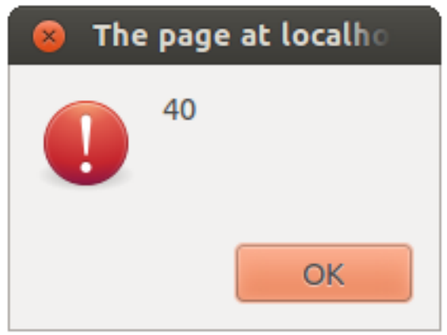
```
<script>
    cljs.repl.init();
    scriptany.register("text/clojurescript", function (text)  {
        var ret = cljs.repl.read_eval_print(text);
        console.log("RTN:", ret);
    });
</script>

<script type="text/clojurescript">
    (defn foo [x] (* 2 x))
</script>

<script type="text/javascript">
    scriptany.barrier();
    alert(cljs.user.foo(20));
</script>

</head>
<body>

    <div id="hello">Hello World!</div>

    <div>Hint: view source</div>

</body>
</html>
~
```

scriptany-cljs.html 24,27 Bot

"scriptany-cljs.html" 35L, 747C [w]

ClojureScript-in-ClojureSc × | ClojureScript-in-ClojureSc × | localhost:8080/scriptany- ×

localhost:8080/scriptany-cljs.html



Elements | Resources | Network | Sources | Timeline | Profiles | Audits | Console

```
RTN: function foo(x){
return cljs.core._STAR_.call(null,2,x);
}                                                    scriptany-cljs.html:14
```

<top frame> | All | Errors | Warnings | Logs | Debug

File Edit View Search Terminal Help

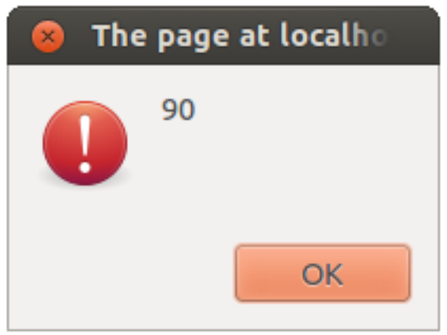
```
        cljs.repl.init();
        scriptany.register("text/clojurescript", function (text)  {
            var ret = cljs.repl.read_eval_print(text);
            console.log("RTN:", ret);
        });
    </script>

    <script type="text/clojurescript">
        (defn foo [x] (* 2 x))
    </script>

    <script type="text/clojurescript" src="bar.cljs">
    </script>

    <script type="text/javascript">
        scriptany.barrier();
        alert(cljs.user.bar(30));
    </script>

</head>
<body>

    <div id="hello">Hello World!</div>

    <div>Hint: view source</div>

</body>
```

scriptany-cljs.html 29,0-1 90%

"scriptany-cljs.html" 38L, 812C [w]

ClojureScript-in-ClojureSc × | ClojureScript-in-ClojureSc × | localhost:8080/scriptany- ×

localhost:8080/scriptany-cljs.html



Elements | Resources | Network | Sources | Timeline | Profiles | Audits | Console

Resource interpreted as Script but transferred with MIME type application/octet-stream: "http://localhost:8080/bar.cljs". scriptany-cljs.html:4

RTN: function foo(x){
return cljs.core._STAR_.call(null,2,x);
}
scriptany-cljs.html:14

▶XHR finished loading: "http://localhost:8080/bar.cljs". scriptany.js:27

print from bar.cljs ambtj.js:131

ambtj.js:131

RTN: null scriptany-cljs.html:14

<top frame> | All | Errors | Warnings | Logs | Debug | 1

Node.js Examples

- ClojureScript REPL (Node.js)

File Edit View Search Terminal Help

```
momote ~/c1 $ ./run.js repl-out/noderepl.js
;; ClojureScript-in-ClojureScript
;;   - http://github.com/kanaka/clojurescript
;;   - A port of the ClojureScript compiler to ClojureScript
cljs.user=>
```

File Edit View Search Terminal Help

```
momote ~/c1 $ ./run.js repl-out/noderepl.js
;; ClojureScript-in-ClojureScript
;;   - http://github.com/kanaka/clojurescript
;;   - A port of the ClojureScript compiler to ClojureScript
cljs.user=> (+ 1 1)
2
cljs.user=>
```

Node.js Examples

- ClojureScript REPL (Node.js)
  - `./run.js repl-out/noderepl.js`
- ClojureScript shebang scripts (Node.js)

File Edit View Search Terminal Help

```
momote ~/c1 $ cat shebang.cljs
#!./run.js repl-out/nodecljs.js

(ns shebang-test)

(def dbl (fn* [x] (* 2 x)))

(println "Hello World UNIX World.")
(println "(dbl 16) is:" (dbl 16))

8
momote ~/c1 $
```

File Edit View Search Terminal Help

```
momote ~/c1 $ cat shebang.cljs
#!./run.js repl-out/nodecljs.js

(ns shebang-test)

(def dbl (fn* [x] (* 2 x)))

(println "Hello World UNIX World.")
(println "(dbl 16) is:" (dbl 16))

8
momote ~/c1 $ ./shebang.cljs
Hello World UNIX World.
(dbl 16) is: 32
momote ~/c1 $
```

File Edit View Search Terminal Help

```
momote ~/c1 $ cat shebang.cljs
#!./run.js repl-out/nodecljs.js

(ns shebang-test)

(def dbl (fn* [x] (* 2 x)))

(println "Hello World UNIX World.")
(println "(dbl 16) is:" (dbl 16))

8
momote ~/c1 $ ./shebang.cljs
Hello World UNIX World.
(dbl 16) is: 32
momote ~/c1 $ ./shebang.clj
Hello World UNIX World.
(dbl 16) is: 32
momote ~/c1 $
```

```
File  Edit  View  Search  Terminal  Help
momote ~/c1 $ cat shebang.cljs
#!./run.js repl-out/nodecljs.js

(ns shebang-test)

(def dbl (fn* [x] (* 2 x)))

(println "Hello World UNIX World.")
(println "(dbl 16) is:" (dbl 16))

8
momote ~/c1 $ ./shebang.cljs
Hello World UNIX World.
(dbl 16) is: 32
momote ~/c1 $ ./shebang.clj
Hello World UNIX World.
(dbl 16) is: 32
momote ~/c1 $ time ./shebang.clj
Hello World UNIX World.
(dbl 16) is: 32

real    0m1.129s
user    0m1.588s
sys     0m0.060s
momote ~/c1 $
```

```
File  Edit  View  Search  Terminal  Help
(ns shebang-test)

(def dbl (fn* [x] (* 2 x)))

(println "Hello World UNIX World.")
(println "(dbl 16) is:" (dbl 16))

8
momote ~/c1 $ ./shebang.cljs
Hello World UNIX World.
(dbl 16) is: 32
momote ~/c1 $ ./shebang.clj
Hello World UNIX World.
(dbl 16) is: 32
momote ~/c1 $ time ./shebang.clj
Hello World UNIX World.
(dbl 16) is: 32

real    0m1.129s
user    0m1.588s
sys     0m0.060s
momote ~/c1 $ time ./shebang.cljs
Hello World UNIX World.
(dbl 16) is: 32

real    0m0.516s
user    0m0.464s
sys     0m0.056s
momote ~/c1 $
```

```
File  Edit  View  Search  Terminal  Help
(dbl 16) is: 32
momote ~/c1 $ time ./shebang.clj
Hello World UNIX World.
(dbl 16) is: 32

real    0m1.129s
user    0m1.588s
sys     0m0.060s
momote ~/c1 $ time ./shebang.cljs
Hello World UNIX World.
(dbl 16) is: 32

real    0m0.516s
user    0m0.464s
sys     0m0.056s
momote ~/c1 $
momote ~/c1 $
momote ~/c1 $ cat shebang.clj
#!/usr/bin/env clj

(ns shebang-test)

(def dbl (fn* [x] (* 2 x)))

(println "Hello World UNIX World.")
(println "(dbl 16) is:" (dbl 16))

8
momote ~/c1 $
```

Node.js Examples

- ClojureScript REPL (Node.js)
  - `./run.js repl-out/noderepl.js`
- ClojureScript shebang scripts (Node.js)
  - `time ./shebang.cljs`
  - `time ./shebang.clj`
- ClojureScript-in-ClojureScript compiler (Node.js)

File Edit View Search Terminal Help

momote ~/c2/node $

File Edit View Search Terminal Help

```
momote ~/c2/node $ mkdir -p out2/cljs
momote ~/c2/node $
```

File Edit View Search Terminal Help

```
momote ~/c2/node $ mkdir -p out2/cljs
momote ~/c2/node $ cp out/goog.js out2/
momote ~/c2/node $ cp out/cljs/core.js out2/cljs
momote ~/c2/node $
```

File Edit View Search Terminal Help

```
momote ~/c2/node $ mkdir -p out2/cljs
momote ~/c2/node $ cp out/goog.js out2/
momote ~/c2/node $ cp out/cljs/core.js out2/cljs
momote ~/c2/node $ ./run.js out/cljsc.js hello.cljs > hello.js
momote ~/c2/node $
```

File Edit View Search Terminal Help

```
momote ~/c2/node $ mkdir -p out2/cljs
momote ~/c2/node $ cp out/goog.js out2/
momote ~/c2/node $ cp out/cljs/core.js out2/cljs
momote ~/c2/node $ ./run.js out/cljsc.js hello.cljs > hello.js
momote ~/c2/node $ ./run2.js hello.js
hello world 256


momote ~/c2/node $
```

File Edit View Search Terminal Help

```
momote ~/c2/node $ mkdir -p out2/cljs
momote ~/c2/node $ cp out/goog.js out2/
momote ~/c2/node $ cp out/cljs/core.js out2/cljs
momote ~/c2/node $ ./run.js out/cljsc.js hello.cljs > hello.js
momote ~/c2/node $ ./run2.js hello.js
hello world 256


momote ~/c2/node $ cat hello.cljs
(ns hello)

(set! cljs.core/*print-fn* (.-log js/console))

(def sqr (fn* [x] (* x x)))

(println "hello world" (sqr 16))
momote ~/c2/node $
```

Node.js Examples

- ClojureScript REPL (Node.js)
  - `./run.js repl-out/noderepl.js`
- ClojureScript shebang scripts (Node.js)
  - `time ./shebang.cljs`
  - `time ./shebang.clj`
- ClojureScript-in-ClojureScript compiler (Node.js)
  - `mkdir -p out2/cljs`
  - `cp out/goog.js out2/`
  - `cp out/cljs/core.cljs # Hack alert`
  - `./run.js out/cljsc.js hello.cljs > hello.js`
  - `./run2.js hello.js`

Clojure-in-Clojure

- The dream of **Clojure-in-Clojure** is a Clojure implementation that is written entirely in Clojure except for a very small platform specific core.
- It would allow easy porting to new platforms (among other advantages).
- **ClojureScript** itself is a step towards this goal; more of the compiler is implemented in Clojure.
- **ClojureScript-in-ClojureScript** is another step towards that goal; more of the compiler is implemented in Clojure/ClojureScript.

ClojureScript -> ClojureScript-in-ClojureScript

ClojureScript-in-ClojureSc × ClojureScript-in-ClojureSc ×

localhost:8080/index.html#/cljs-overview

ClojureScript -> ClojureScript-in-ClojureScript

Standard ClojureScript

Input

Program: ClojureScript.clj

Runtime: JVM

ClojureScript-in-ClojureSc × ClojureScript-in-ClojureSc ×

localhost:8080/index.html#/cljs-overview

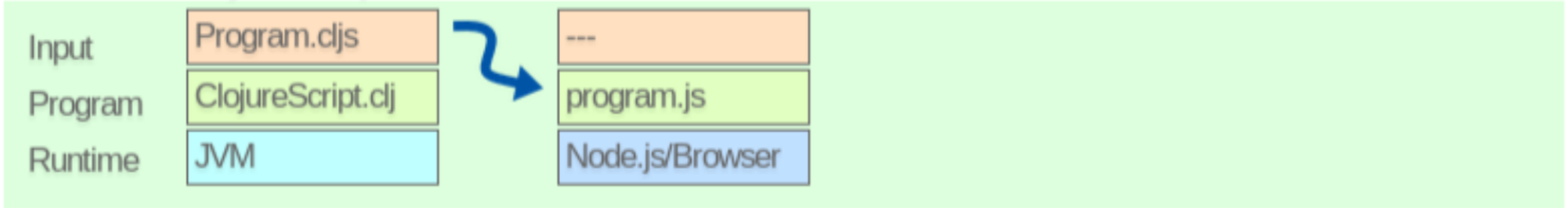
ClojureScript -> ClojureScript-in-ClojureScript

Standard ClojureScript

Input	Program.cljs
Program	ClojureScript.clj
Runtime	JVM

ClojureScript -> ClojureScript-in-ClojureScript

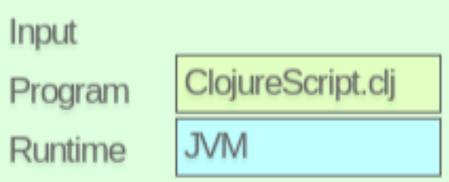
Standard ClojureScript



ClojureScript -> ClojureScript-in-ClojureScript

Standard ClojureScript

ClojureScript-in-ClojureScript



ClojureScript-in-ClojureSc × ClojureScript-in-ClojureSc ×

localhost:8080/index.html#/cljs-overview

ClojureScript -> ClojureScript-in-ClojureScript

Standard ClojureScript

Input	Program.cljs	→	---
Program	ClojureScript.clj	→	program.js
Runtime	JVM	→	Node.js/Browser

ClojureScript-in-ClojureScript

Input	ClojureScript.cljs
Program	ClojureScript.clj
Runtime	JVM

ClojureScript -> ClojureScript-in-ClojureScript

Standard ClojureScript

Input	Program.cljs	→	---
Program	ClojureScript.clj	→	program.js
Runtime	JVM	→	Node.js/Browser

ClojureScript-in-ClojureScript

Input	ClojureScript.cljs	→	
Program	ClojureScript.clj	→	ClojureScript.js
Runtime	JVM	→	Node.js/Browser

ClojureScript -> ClojureScript-in-ClojureScript

Standard ClojureScript

Input	Program.cljs	→	---
Program	ClojureScript.clj	→	program.js
Runtime	JVM	→	Node.js/Browser

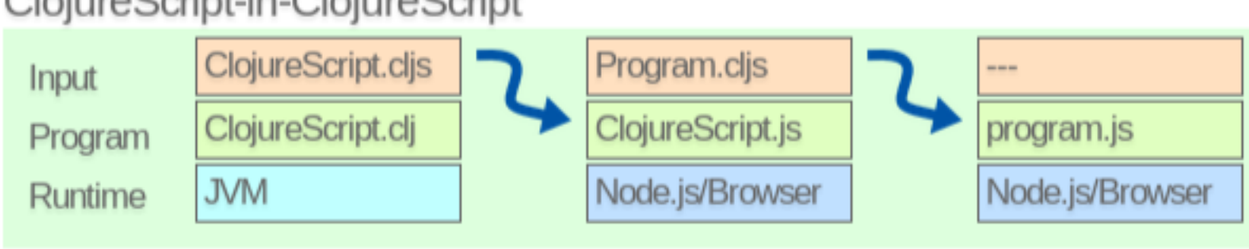
ClojureScript-in-ClojureScript

Input	ClojureScript.cljs	→	Program.cljs
Program	ClojureScript.clj	→	ClojureScript.js
Runtime	JVM	→	Node.js/Browser

ClojureScript -> ClojureScript-in-ClojureScript

Standard ClojureScript

ClojureScript-in-ClojureScript



ClojureScript -> ClojureScript-in-ClojureScript

Standard ClojureScript

Input	Program.cljs	→	---
Program	ClojureScript.clj	→	program.js
Runtime	JVM		Node.js/Browser

ClojureScript-in-ClojureScript

Input	ClojureScript.cljs	→	Program.cljs	→	---
Program	ClojureScript.clj	→	ClojureScript.js	→	program.js
Runtime	JVM		Node.js/Browser		Node.js/Browser

Self-hosted ClojureScript-in-ClojureScript

Input	
Program	ClojureScript.clj
Runtime	JVM

ClojureScript -> ClojureScript-in-ClojureScript

Standard ClojureScript

Input	Program.cljs	→	---
Program	ClojureScript.clj	→	program.js
Runtime	JVM	→	Node.js/Browser

ClojureScript-in-ClojureScript

Input	ClojureScript.cljs	→	Program.cljs	→	---
Program	ClojureScript.clj	→	ClojureScript.js	→	program.js
Runtime	JVM	→	Node.js/Browser	→	Node.js/Browser

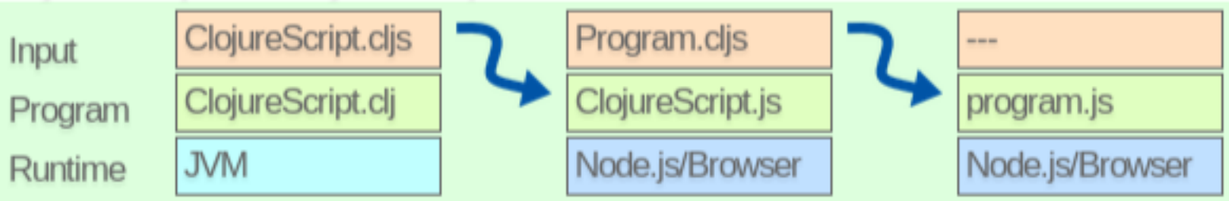
Self-hosted ClojureScript-in-ClojureScript

Input	ClojureScript.cljs
Program	ClojureScript.clj
Runtime	JVM

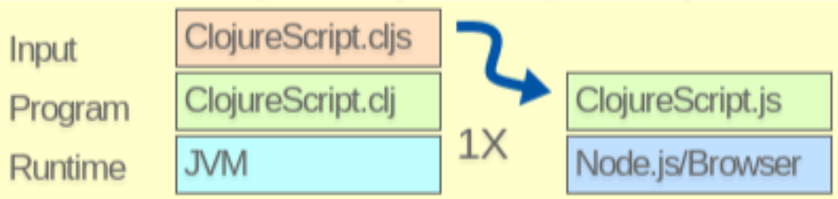
ClojureScript -> ClojureScript-in-ClojureScript

Standard ClojureScript

ClojureScript-in-ClojureScript



Self-hosted ClojureScript-in-ClojureScript



ClojureScript -> ClojureScript-in-ClojureScript

Standard ClojureScript

Input	Program.cljs	→	---
Program	ClojureScript.clj	→	program.js
Runtime	JVM	→	Node.js/Browser

ClojureScript-in-ClojureScript

Input	ClojureScript.cljs	→	Program.cljs	→	---
Program	ClojureScript.clj	→	ClojureScript.js	→	program.js
Runtime	JVM	→	Node.js/Browser	→	Node.js/Browser

Self-hosted ClojureScript-in-ClojureScript

Input	ClojureScript.cljs	→	ClojureScript.cljs	
Program	ClojureScript.clj	→	ClojureScript.js	
Runtime	JVM	1X	Node.js/Browser	

ClojureScript -> ClojureScript-in-ClojureScript

Standard ClojureScript

Input	Program.cljs	→	---
Program	ClojureScript.clj	→	program.js
Runtime	JVM		Node.js/Browser

ClojureScript-in-ClojureScript

Input	ClojureScript.cljs	→	Program.cljs	→	---
Program	ClojureScript.clj	→	ClojureScript.js	→	program.js
Runtime	JVM		Node.js/Browser		Node.js/Browser

Self-hosted ClojureScript-in-ClojureScript

Input	ClojureScript.cljs	→	ClojureScript.cljs	→	
Program	ClojureScript.clj	1X →	ClojureScript.js	→	ClojureScript.js
Runtime	JVM		Node.js/Browser		Node.js/Browser

ClojureScript -> ClojureScript-in-ClojureScript

Standard ClojureScript

Input	Program.cljs	→	---
Program	ClojureScript.clj	→	program.js
Runtime	JVM		Node.js/Browser

ClojureScript-in-ClojureScript

Input	ClojureScript.cljs	→	Program.cljs	→	---
Program	ClojureScript.clj	→	ClojureScript.js	→	program.js
Runtime	JVM		Node.js/Browser		Node.js/Browser

Self-hosted ClojureScript-in-ClojureScript

Input	ClojureScript.cljs	→	ClojureScript.cljs	→	Program.cljs
Program	ClojureScript.clj	→	ClojureScript.js	→	ClojureScript.js
Runtime	JVM	1X	Node.js/Browser		Node.js/Browser

ClojureScript -> ClojureScript-in-ClojureScript

Standard ClojureScript

Input	Program.cljs	→	---
Program	ClojureScript.clj	→	program.js
Runtime	JVM	→	Node.js/Browser

ClojureScript-in-ClojureScript

Input	ClojureScript.cljs	→	Program.cljs	→	---
Program	ClojureScript.clj	→	ClojureScript.js	→	program.js
Runtime	JVM	→	Node.js/Browser	→	Node.js/Browser

Self-hosted ClojureScript-in-ClojureScript

Input	ClojureScript.cljs	→	ClojureScript.cljs	→	Program.cljs	→	---
Program	ClojureScript.clj	→	ClojureScript.js	→	ClojureScript.js	→	program.js
Runtime	JVM	1X	Node.js/Browser	→	Node.js/Browser	→	Node.js/Browser

Porting Steps

A chronological rather than functional overview of the process.

Review: ClojureScript Compilation Process

Review: ClojureScript Compilation Process

- Read
  - symbols, comments, quote, metadata, syntax quote and unquote
  - literals: characters, maps, lists, vectors, keywords, strings
  - dispatch reader (#...)
    - literals: sets, regex, anonymous functions
    - discard/ignore, quoted vars

Review: ClojureScript Compilation Process

- Read
  - symbols, comments, quote, metadata, syntax quote and unquote
  - literals: characters, maps, lists, vectors, keywords, strings
  - dispatch reader (#...)
    - literals: sets, regex, anonymous functions
    - discard/ignore, quoted vars
- Analyze
  - specials expansion/parsing (def, fn*, if, let*, recur, etc)
  - macro expansion/parsing
  - regular form parsing

Review: ClojureScript Compilation Process

- Read
  - symbols, comments, quote, metadata, syntax quote and unquote
  - literals: characters, maps, lists, vectors, keywords, strings
  - dispatch reader (#...)
    - literals: sets, regex, anonymous functions
    - discard/ignore, quoted vars
- Analyze
  - specials expansion/parsing (def, fn*, if, let*, recur, etc)
  - macro expansion/parsing
  - regular form parsing
- Compile/emit
  - call analyzer to build AST
  - resolve (compile or copy) dependencies
    - Closure lib, clojure.*, cljs.core, etc
  - traverse AST emitting JavaScript
  - call Closure Compiler (optimize, merge)

Review: ClojureScript Compilation Process

- Read
  - symbols, comments, quote, metadata, syntax quote and unquote
  - literals: characters, maps, lists, vectors, keywords, strings
  - dispatch reader (#...)
    - literals: sets, regex, anonymous functions
    - discard/ignore, quoted vars
- Analyze
  - specials expansion/parsing (def, fn*, if, let*, recur, etc)
  - macro expansion/parsing
  - regular form parsing
- Compile/emit
  - call analyzer to build AST
  - resolve (compile or copy) dependencies
    - Closure lib, clojure.*, cljs.core, etc
  - traverse AST emitting JavaScript
  - call Closure Compiler (optimize, merge)

- Eval
  - at JavaScript load time

ClojureScript-in-ClojureSc × | ClojureScript-in-ClojureSc ×

localhost:8080/index.html#/porting1

Step 1. Minimal web REPL

The beginning: a Clojure/conj hackathon with chouser.

- Copy src/clj/cljs/{compiler,analyzer}.clj to src/cljs/cljs/*.cljs
- Static snapshot of the analyzer namespaces atom.
- Move macros out and pull in using :use-macros
- Stub out a bunch of stuff
- Rewrite emit-constant multimethods as EmitConstant protocol (ClojureScript protocol oriented rather than type/class oriented).
- Bootstrap HTML/JavaScript file:
  - load the static namespaces snapshot
  - setup the analyzer environment
- **RESULT**:
  - very simple web REPL
  - invoke cljs.core functions
  - specials (fn*, if, def, let*)
  - no macros (defn, let, etc)

Step

- A
- n
- R

Step 2. Node.js REPL Support

- Add Node.js file loading to Google Closure (goog.js)
- node.js launch script (run.js):
  - load the static namespaces snapshot
  - provide a basic REPL using Node.js readline module.
- **RESULT**:
  - very simple Node.js REPL
  - equivalent to the simple web REPL

Ste

- T
  fo
- L
  c
- N
- C

- R

Step 3. Dynamic namespaces data

- The namespaces atom: the analyzer's namespace and var data for def forms being compiled.
- Lookup vars during compilation (only macros are resolved to compiler's namespace vars).
- Normally discarded after compilation, but cljs-in-cljs needs it!
- Old and busted:
  - Write entire namespaces content to a file
  - Manually add "(set! cljs.analyzer.namespaces DATA)"
- **RESULT** (New hotness):
  - Emit namespaces data to end of each compiled file
  - Just for the namespace being compiled
  - Swapped into to build up the namespaces as each file is loaded.

Ste

- P
- P
- A
- R

Step 4. Complete the reader

- Port syntax quote/unqoute from Java code. Syntax quote/unquote is a key component of macros.
- Port the anonymous function and function argument reader
- Activate comment reader
- **RESULT**:
  - Full reader (modulo bugs)

Ste

- d
- Ir
  is
- P
- R

Step 5. defmacro

- defmacro is itself a macro. This leads to some challenges.
- In ClojureScript, macros are defined on the Clojure/JVM side. Only the expanded result is emitted as JS code.
- Port defmacro macro from into cljs-in-cljs core.cljs file.
- **RESULT**:
  - Runtime definition of new macros.
  - No ahead-of-time definition of macros due to the split between the Clojure vars namespace and the ClojureScript compiled namespaces.
  - Still no predefined core macros.

result

ars

Step 6. clj-defmacro and core macros

- Add clj-defmacro to core_macros.clj and pull it in using :use-macros so that it is a Clojure (JVM) side version of defmacro.
- Use clj-defmacro to port macros from Clojure core.clj and ClojureScript core.cljs into cljs-in-cljs core.cljs file.
- Port destructure function for use by let, loop, fn macros.
- **RESULT**:
  - Much of core Clojure is now available
  - Starting to feel like a real REPL.

Step 7. Types, protocols and first-class symbols

- Definition and extension of protocols uses metadata on symbols to pass around information.
- In ClojureScript, symbols are just prefixed strings.
- Problem: no custom attributes on strings in JS.
- Change symbols to real things that support metadata.
- Port reify, extend-type, deftype, defrecord, and defprotocol macros and supporting functions from ClojureScript core.clj. Port extend-protocol from Clojure core.clj.
- **RESULT**:
  - New types, protocols, records, etc can be defined at "runtime" (more specifically at cljs-in-cljs compile time).

Step 8. Namespaces as a value

- In Clojure, the *ns* symbol refers to the mutable Namespace object of the current namespace.
- The namespaces data in ClojureScript is a normal Clojure persistent datastructure in an atom.
- Problem: modifying a namespace in the namespaces data (via a def*) will not automatically update the copy of it in *ns*.
- Drop *ns* for cljs-in-cljs and replace with *ns-sym*: a dynamic "var" with the namespace symbol.
- Update *ns* uses to use *ns-sym* and deref namespaces atom as appropriate.
- Port some namespace related functions
- **RESULT**:
  - No stale *ns*.
  - Namespace creation and other functions.
  - More correct macro and symbol resolution.

Ste

- I
- I
- U
- R

Step 9. File I/O Support

- Implement Java-like File object.
- Implement Java-like I/O readers/streams (maybe).
- Use node.js file read/write routines. Use XMLHttpRequest for web file read.
- **RESULT**:
  - load-file now works.
  - Testing via direct cljs file loading rather than pasted forms.

Step 10. Miscellaneous

- Port defmulti and defmethod macros.
- Output "streams" (*out*, *err*, *rtn*)
  - Rendering functions rather than file streams
  - *rtn* for rendering of return value
- Port closure.clj
- Line numbering pushbackreader
- Bug fixes: / symbol, keywords, regexes, unicode, etc, etc
- Tests
- REPL functionality/cleanup
- **RESULT**:
  - Useful enough to hack up cool examples for a Clojure conference.

Ste

On go

Most t
intera

The h
compi

Step 11. Self-hosting

On going work

Most things compile except for some tricky macro/var/namespace interactions in core.clj/core.cljs

The hack: copy the already compiled core.js file and the rest will compile.

ClojureScript-in-ClojureSc ×
ClojureScript-in-ClojureSc ×

localhost:8080/index.html#/finish

Finishing Up

Future Work

- Require/Use/CLASSPATH issues
- Enhance debug repl
  - Vision: like firebug but for (and in) ClojureScript
  - Make it a compile/lein option
- Self-hosting (get all of core.clj to compile)
- Merging upstream ClojureScript (branched in Nov 2012)
- Push to upstream ClojureScript
- Greater use of Node.js
  - npmjs.org is the fastest growing language module repository
  - Create and push Node.js package (npm)
  - Node.js library interop
- Performance work
  - V8 profiling
  - asm.js: asmjs.org (not cljs-in-cljs specific)
- Cool new applications
  - cljs-in-cljs + catnip + js-git
  - LightTable Light (no backend needed)
  - All the cool things I can't think of, that all of you will create.

Connect:

- @bus_kanaka
- @noVNC (noVNC & HTML5 related)
- github.com/kanaka
- blog.n01se.net/kanaka
- cljs@martintribe.org
- IRC discussion: #cljs-in-cljs (freenode)

Links:

- This presentation: kanaka.github.com/cljs-in-cljs-presentation
  - WARNING: very HTML5 heavy.
- Source for presentation: github.com/kanaka/cljs-in-cljs-presentation
- ClojureScript-in-ClojureScript Project: github.com/kanaka/clojurescript
- Upstream ClojureScript: github.com/kanaka/clojurescript
- ClojureScript-in-ClojureScript web REPL: ClojureScript.net
- Scriptany: github.com/kanaka/Scriptany
- Wisp - From-scratch Clojure implementation on JavaScript: github.com/Gozala/wisp

ClojureScript-in-ClojureSc × ClojureScript-in-ClojureSc ×

localhost:8080/index.html#/thanks

Thanks:

- Chouser: the initial bootstrap and other help.
- Fogus: Himera design used in ClojureScript.net
- Irakli Gozalishvili: bug fixes
- Arthur Edelstein: jq-console/editor for ClojureScript.net
- Mike Anderson: surprise-ending example.
- Game of Life demo: https://github.com/sw1nn
- David Nolen for saying "Wait, this isn't right" ... there really is no better encouragement than that.
- Alan Dipert for thinking (and saying repeatedly) that this is cool.
- Aaron Brooks for being a sounding board.
- Supporting colleagues at LonoCloud
- Alex Miller for putting on an awesome conference.
- Others I've forgotten

ClojureScript-in-ClojureSc × | ClojureScript-in-ClojureSc ×

localhost:8080/index.html#/thanks

Thanks:

- Chouser: the initial bootstrap and other help.
- Fogus: Himera design used in ClojureScript.net
- Irakli Gozalishvili: bug fixes
- Arthur Edelstein: jq-console/editor for ClojureScript.net
- Mike Anderson: surprise-ending example.
- Game of Life demo: https://github.com/sw1nn
- David Nolen for saying "Wait, this isn't right" ... there really is no better encouragement than that.
- Alan Dipert for thinking (and saying repeatedly) that this is cool.
- Aaron Brooks for being a sounding board.
- Supporting colleagues at LonoCloud
- Alex Miller for putting on an awesome conference.
- Others I've forgotten

Questions?