Kevin Lynagh
Keming Labs

ClojureScript
JavaScript

1.

Clojure is **Simpler** than JavaScript

2.

Using JavaScript from ClojureScript

is *MOAR*

Fun!

than using JavaScript from JavaScript.

2. Using JavaScript from ClojureScript

is *MOAR*

Safe!

than using JavaScript from JavaScript.

2. Using JavaScript from ClojureScript

is *MOAR*

Extendable!

than using JavaScript from JavaScript.

3.

The trip from

.js

to

.cljs is an

Adventure!

(you might get hurt)

0. story

Imperial

∷

Metric

JavaScript

::

Clojure

JavaScript

::

::

Clojure

JavaScript

::

::

Clojure

Lets
talk
about

JS

JS has a lot of **stupid** in it

News at 11.

— Brendan Eich (creator of JavaScript)

Attack of the *killer* Globals

```
var x = 12;
```

Attack of the *killer* Globals

```
x = 12;
```

Clojure Solution

Things

Clojure
Solution

Things don't

Clojure Solution

Things don’t change

Clojure Solution

JavaScriptNamespaces

JavaScriptNamespaces

```
(function(){


})();
```

```
(function( $ ){

  var methods = {
    init: function( options ) {},
    show: function( ) {},
    hide: function( ) {},
    update: function( content ) {}
  };

  $.fn.tooltip = function( method ) {
    if ( methods[method] ) {
      return methods[method].apply(
        this, Array.prototype.slice.call( arguments, 1 ));
    } else if ( typeof method === 'object' || ! method ) {
      return methods.init.apply( this, arguments );
    } else {
      $.error( 'Method ' +  method + ' does not exist!' );
    }
  };

})( jQuery );
```

This is

Crazy

( )

(CoffeeScript *will not* save you)

Clojure Solution

Use

Clojure
Solution

Use

Name-

Clojure Solution

Clojure Solution

Use Name-spaces

ClojureNamespaces

```
;;src/clj/my_stuff.clj
(ns my-stuff)

(defn thing [x] )
(defn another [x y] )
```

ClojureNamespaces

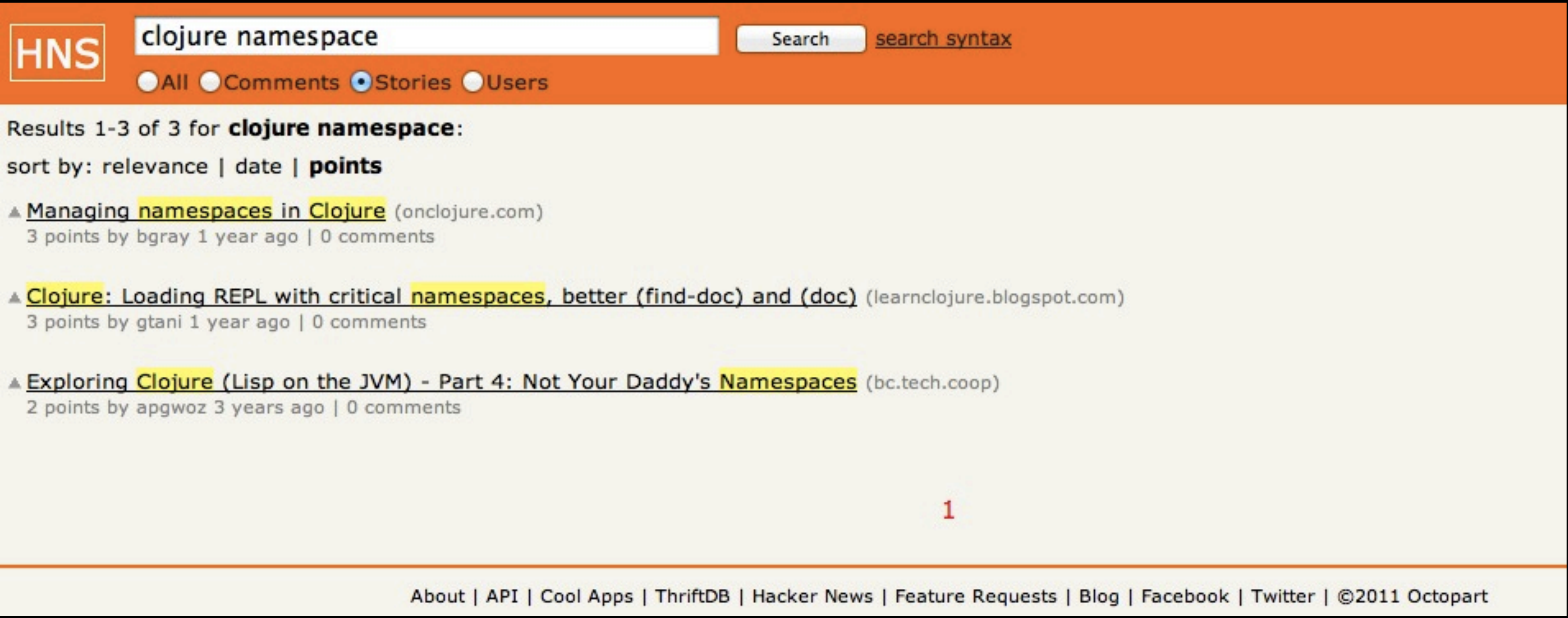
```
;;src/clj/elsewhere.clj
(ns elsewhere)
```

ClojureNamespaces

```clojure
;;src/clj/elsewhere.clj
(ns elsewhere
  (:use [my-stuff :only [thing]]))
```

ClojureNamespaces

```clojure
;;src/clj/elsewhere.clj
(ns elsewhere
  (:use [my-stuff :only [thing]
         :rename {thing other-thing}]))
```


HNS

clojure namespace | Search | search syntax

All Comments [✓ Stories] Users

Results 1-3 of 3 for **clojure namespace**:

sort by: relevance | date | **points**

▲ Managing namespaces in Clojure (onclojure.com)
3 points by bgray 1 year ago | 0 comments

▲ Clojure: Loading REPL with critical namespaces, better (find-doc) and (doc) (learnclojure.blogspot.com)
3 points by gtani 1 year ago | 0 comments

▲ Exploring Clojure (Lisp on the JVM) - Part 4: Not Your Daddy's Namespaces (bc.tech.coop)
2 points by apgwoz 3 years ago | 0 comments

1

About | API | Cool Apps | ThriftDB | Hacker News | Feature Requests | Blog | Facebook | Twitter | ©2011 Octopart

Recap

1.

Clojure

is

Simpler

than

JavaScript

Only

Geniuses

use

functional

languages

Only
Geniuses
use
functional
languages

Myth

2. Using JavaScript from ClojureScript

InteropReview

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

InteropReview

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

Interop**Review**

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

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

ADifference

```
(.aMethod js/anObject)
```

A Difference

```
(.aMethod js/anObject)

(. js/anObject (aMethod))
```

Types

Types

Scalars

Types

Scalars

Native

Types

Scalars

Native

Collections

Types

Scalars

Native

Collections

Foreign (lazy in Clojure)

JS array <=> Cljs vector

```
(array 1 2 3) ;;=> [1,2,3];
(apply array [1 2 3]) ;;=> [1,2,3];

(cljs.core.Vector/fromArray js/x)
  ;;=> [1 2 3]
```

JS Objects

```
(let [o (js-obj)]
  (aset o "x" 1)
  (set! (.y o) 2)
  o)
;;=> {"x":1,"y":2};
```

Reader Literals

```
#js[1 2 3]

#js{"key1" "val1",
    "key2" "val2"}
```

2. Using JavaScript from ClojureScript

is *MOAR* **Extendable!**

than using JavaScript from JavaScript.

```
jQuery.select("#main")
  .append("<span>")
  .text("hello");
```

```
jQuery.select("#main")
  .append("<span>")
  .text("hello");

(-> js/jQuery
    (.select "#main")
    (.append "<span>")
    (.text "hello"))
```

```
jQuery.select("#main")
  .append("<span>")
  .text("hello");

(-> js/jQuery
    (.select "#main")
    (my-append "<span>")
    (.text "hello"))
```

Compare JavaScript

Compare JavaScript

```
var $main = jQuery.select("#main");
my_appender($main, "<span>");
$main.text("hello");
```

Compare JavaScript

```
var $main = jQuery.select("#main");
my_appender($main, "<span>");
$main.text("hello");


jQuery.fn.my_appender = function(x){};
```

Build façades

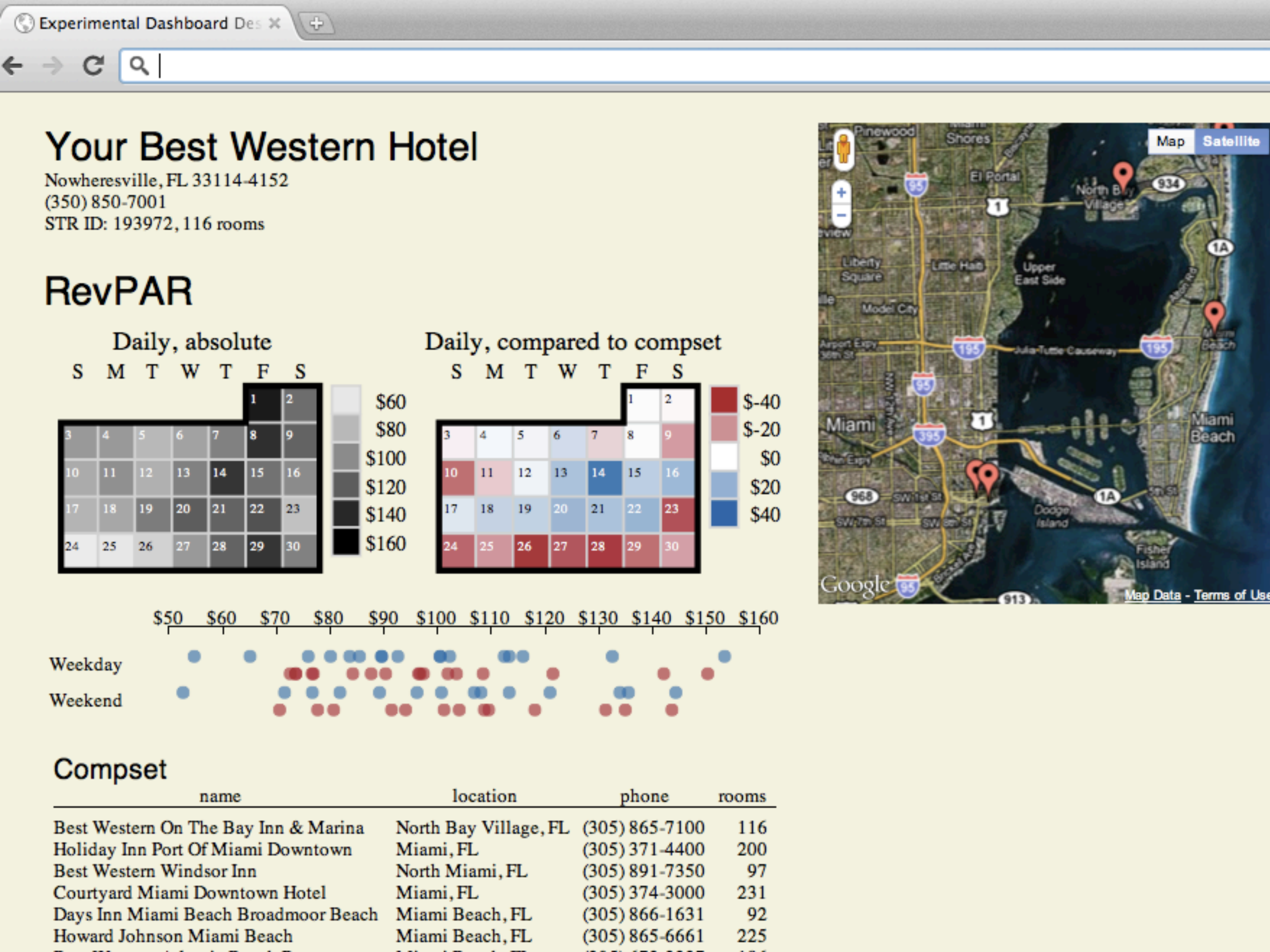
Build façades

```
(defn append [sel node-type]
  (.append sel node-type))

(defn select [sel selector]
  (.select sel selector))

(-> js/jQuery
    (select "#main")
    (append "svg"))
```

façades

decouple

intent *implementation*



Your Best Western Hotel

Nowheresville, FL 33114-4152
(350) 850-7001
STR ID: 193972, 116 rooms

RevPAR

Daily, absolute

Daily, compared to compset

Compset

name	location	phone	rooms
Best Western On The Bay Inn & Marina	North Bay Village, FL	(305) 865-7100	116
Holiday Inn Port Of Miami Downtown	Miami, FL	(305) 371-4400	200
Best Western Windsor Inn	North Miami, FL	(305) 891-7350	97
Courtyard Miami Downtown Hotel	Miami, FL	(305) 374-3000	231
Days Inn Miami Beach Broadmoor Beach	Miami Beach, FL	(305) 866-1631	92
Howard Johnson Miami Beach	Miami Beach, FL	(305) 865-6661	225

2.

Recap

Using
JavaScript
from
ClojureScript

is

than using JavaScript from JavaScript.

2.

Recap

Using JavaScript from ClojureScript

is *MOAR*

Fun!

than using JavaScript from JavaScript.

2.

Recap

Using **JavaScript** from **ClojureScript**

is *MOAR* **Safe!**

than using JavaScript from JavaScript.

2.

Recap

Using **JavaScript** from **ClojureScript**

is *MOAR* **Extendable!**

than using JavaScript from JavaScript.

3.

The trip from **.js** to **.cljs** is an *Adventure!*

(you might get hurt)

3.

What we thought we knew until *ClojureScript* showed us otherwise

Lets talk about

this

this this this this

```
var f = function(){ return this; };
```

this this this this

```
var f = function(){ return this; };
f(); //=> DOMWindow
```

this this this this

```
var f = function(){ return this; };
f(); //=> DOMWindow

var dog = {};
dog.name = "Rex";
dog.bark = function(){
  return "My name is " + this.name;
};
```

this this this this

```
var f = function(){ return this; };
f(); //=> DOMWindow

var dog = {};
dog.name = "Rex";
dog.bark = function(){
  return "My name is " + this.name;
};

dog.bark(); // => "My name is Rex"
```

Bewareapply

Bewareapply

```
(apply (. js/dog bark) 1 2 3)
```

Beware apply

```
(apply (. js/dog bark) 1 2 3)
```

```
cljs.core.apply.call(
  null,
  dog.bark,
  cljs.core.Vector.fromArray([1,2,3]));
```

Referential transparency

You Want It To Be One Way. But It's The Other Way.

Recap

1.

Clojure is **Simpler** than JavaScript

2.

Recap

Using **JavaScript** from **ClojureScript**

is *MOAR* **Fun!**

than using JavaScript from JavaScript.

2.

Recap

Using
JavaScript
from
ClojureScript

is *MOAR*
Safe!

than using JavaScript from JavaScript.

2.

Recap

Using JavaScript from ClojureScript

is *MOAR*

Extendable!

than using JavaScript from JavaScript.

3.

Recap

The trip from

.js

to

.cljs is an

Adventure!

(you might get hurt)

4.

A free piece
of
unsolicited

Advice:

Tools don’t

make

the **artist**

Kevin Lynagh
Keming Labs

slides, notes, &c.
keminglabs.com/talks