A Bird's Eye View of

ClojureScript

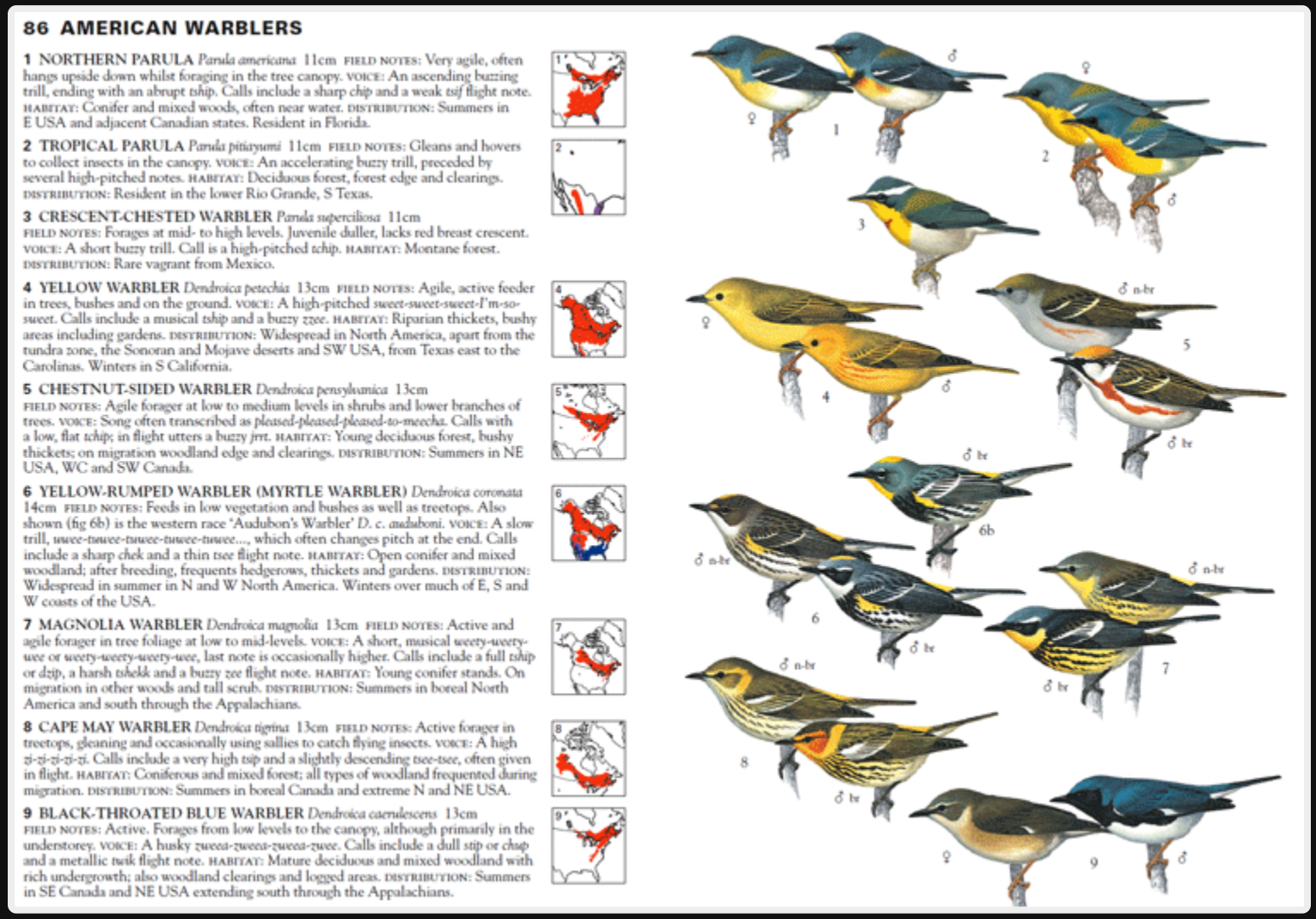
Chandu Tennety

{:github "tennety" :twitter "tennety"}

John Andrews

{:github "jxa" :twitter "xandrews"}

Motivation



86 AMERICAN WARBLERS

1 NORTHERN PARULA *Parula americana* 11cm FIELD NOTES: Very agile, often hangs upside down whilst foraging in the tree canopy. VOICE: An ascending buzzing trill, ending with an abrupt *tship*. Calls include a sharp *chip* and a weak *tsif* flight note. HABITAT: Conifer and mixed woods, often near water. DISTRIBUTION: Summers in E USA and adjacent Canadian states. Resident in Florida.

2 TROPICAL PARULA *Parula pitiayumi* 11cm FIELD NOTES: Gleans and hovers to collect insects in the canopy. VOICE: An accelerating buzzy trill, preceded by several high-pitched notes. HABITAT: Deciduous forest, forest edge and clearings. DISTRIBUTION: Resident in the lower Rio Grande, S Texas.

3 CRESCENT-CHESTED WARBLER *Parula superciliosa* 11cm FIELD NOTES: Forages at mid- to high levels. Juvenile duller, lacks red breast crescent. VOICE: A short buzzy trill. Call is a high-pitched *tchip*. HABITAT: Montane forest. DISTRIBUTION: Rare vagrant from Mexico.

4 YELLOW WARBLER *Dendroica petechia* 13cm FIELD NOTES: Agile, active feeder in trees, bushes and on the ground. VOICE: A high-pitched *sweet-sweet-sweet-I'm-so-sweet*. Calls include a musical *tship* and a buzzy *zzee*. HABITAT: Riparian thickets, bushy areas including gardens. DISTRIBUTION: Widespread in North America, apart from the tundra zone, the Sonoran and Mojave deserts and SW USA, from Texas east to the Carolinas. Winters in S California.

5 CHESTNUT-SIDED WARBLER *Dendroica pensylvanica* 13cm FIELD NOTES: Agile forager at low to medium levels in shrubs and lower branches of trees. VOICE: Song often transcribed as *pleased-pleased-pleased-to-meecha*. Calls with a low, flat *tchip*; in flight utters a buzzy *jrrt*. HABITAT: Young deciduous forest, bushy thickets; on migration woodland edge and clearings. DISTRIBUTION: Summers in NE USA, WC and SW Canada.

6 YELLOW-RUMPED WARBLER (MYRTLE WARBLER) *Dendroica coronata* 14cm FIELD NOTES: Feeds in low vegetation and bushes as well as treetops. Also shown (fig 6b) is the western race 'Audubon's Warbler' *D. c. auduboni*. VOICE: A slow trill, *uwee-tuwee-tuwee-tuwee-tuwee...*, which often changes pitch at the end. Calls include a sharp *chek* and a thin *tsee* flight note. HABITAT: Open conifer and mixed woodland; after breeding, frequents hedgerows, thickets and gardens. DISTRIBUTION: Widespread in summer in N and W North America. Winters over much of E, S and W coasts of the USA.

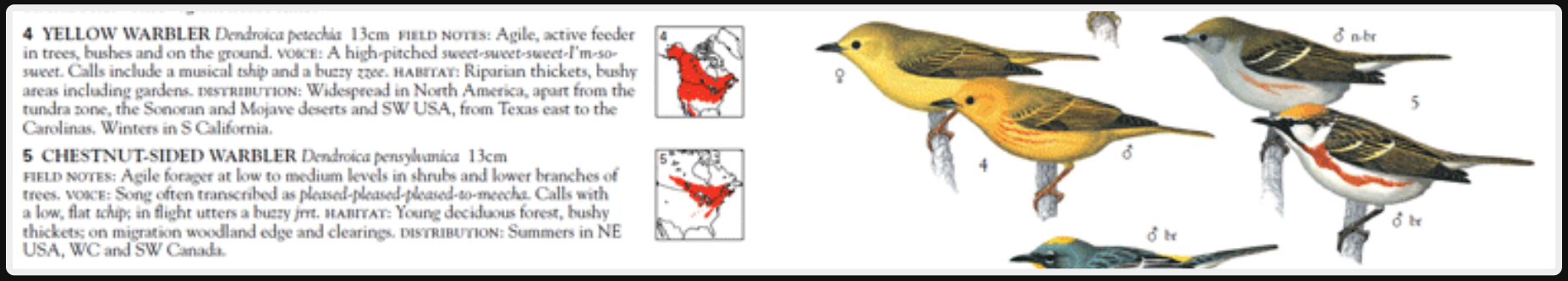
7 MAGNOLIA WARBLER *Dendroica magnolia* 13cm FIELD NOTES: Active and agile forager in tree foliage at low to mid-levels. VOICE: A short, musical *weety-weety-wee* or *weety-weety-weety-wee*, last note is occasionally higher. Calls include a full *tship* or *dzip*, a harsh *tshekk* and a buzzy *zee* flight note. HABITAT: Young conifer stands. On migration in other woods and tall scrub. DISTRIBUTION: Summers in boreal North America and south through the Appalachians.

8 CAPE MAY WARBLER *Dendroica tigrina* 13cm FIELD NOTES: Active forager in treetops, gleaning and occasionally using sallies to catch flying insects. VOICE: A high *zi-zi-zi-zi-zi*. Calls include a very high *tsip* and a slightly descending *tsee-tsee*, often given in flight. HABITAT: Coniferous and mixed forest; all types of woodland frequented during migration. DISTRIBUTION: Summers in boreal Canada and extreme N and NE USA.

9 BLACK-THROATED BLUE WARBLER *Dendroica caerulescens* 13cm FIELD NOTES: Active. Forages from low levels to the canopy, although primarily in the understorey. VOICE: A husky *zweea-zweea-zweea-zwee*. Calls include a dull *stip* or *chup* and a metallic *twik* flight note. HABITAT: Mature deciduous and mixed woodland with rich undergrowth; also woodland clearings and logged areas. DISTRIBUTION: Summers in SE Canada and NE USA extending south through the Appalachians.

Source: Princeton University

Motivation



4 YELLOW WARBLER *Dendroica petechia* 13cm FIELD NOTES: Agile, active feeder in trees, bushes and on the ground. VOICE: A high-pitched *sweet-sweet-sweet-I'm-so-sweet*. Calls include a musical *tship* and a buzzy *zzee*. HABITAT: Riparian thickets, bushy areas including gardens. DISTRIBUTION: Widespread in North America, apart from the tundra zone, the Sonoran and Mojave deserts and SW USA, from Texas east to the Carolinas. Winters in S California.

5 CHESTNUT-SIDED WARBLER *Dendroica pensylvanica* 13cm
FIELD NOTES: Agile forager at low to medium levels in shrubs and lower branches of trees. VOICE: Song often transcribed as *pleased-pleased-pleased-to-meecha*. Calls with a low, flat *tchip*; in flight utters a buzzy *jrrt*. HABITAT: Young deciduous forest, bushy thickets; on migration woodland edge and clearings. DISTRIBUTION: Summers in NE USA, WC and SW Canada.

Yellow Warbler

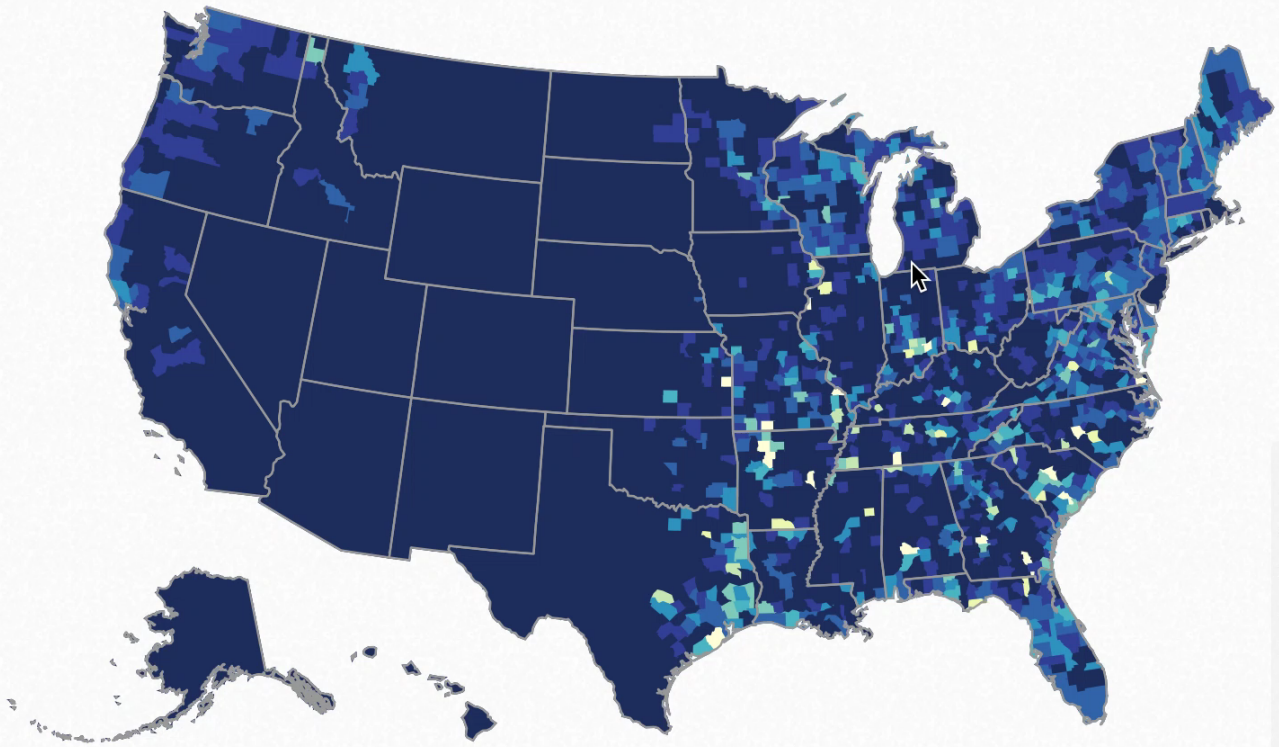
Chestnut-sided Wabler

Demo

BIRDWAVE VISUALIZING BIRD MIGRATIONS

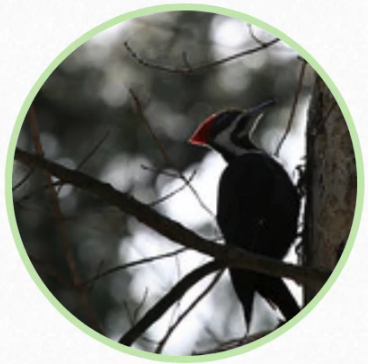
Pileated Woodpecker

December January February March April May June July August September October November



4.4
3.9
3.3
2.8
2.2
1.7
1.1
0.6

Scale represents the ratio of birds sighted to number of sightings
eBird Basic Dataset. Version: EBD_relNov-2013. Cornell Lab of Ornithology, Ithaca, New York. November 2013.



search for species

Abert's Towhee
Acadian Flycatcher
Acanthis/Spinus sp.
Accipiter sp.
Acorn Woodpecker
Acorn Woodpecker (Acorn)
African Collared-Dove
African Collared-Dove (Domestic type or Ringed Turtle-Dove)
African Silverbill
Akekee

1:16

How this works

Data for the App

eBird data set

- Data for 1 year for the US region
- 11 GB of tab-separated values
- Over 1700 species

Data for the App

The need for an API

- Safe, quick data import
- Too much data to load at once
- Dynamic nature of the app
- d3 handles JSON requests

Clojure Syntax

```javascript
/* JavaScript */
function greet(who, event) {
  return "Greetings " + who + "! Welcome to " + event "!";
}

greet("Good People", "LambdaConf 2015");
```

```clojure
;; Clojure
(defn greet [who event]
  (str "Greetings " who "! Welcome to " event "!"))

(greet "Good People" "LambdaConf")
```

Clojure Syntax

- `( )` List
- `[ ]` Vector
- `{ }` Map

Parsing the Data

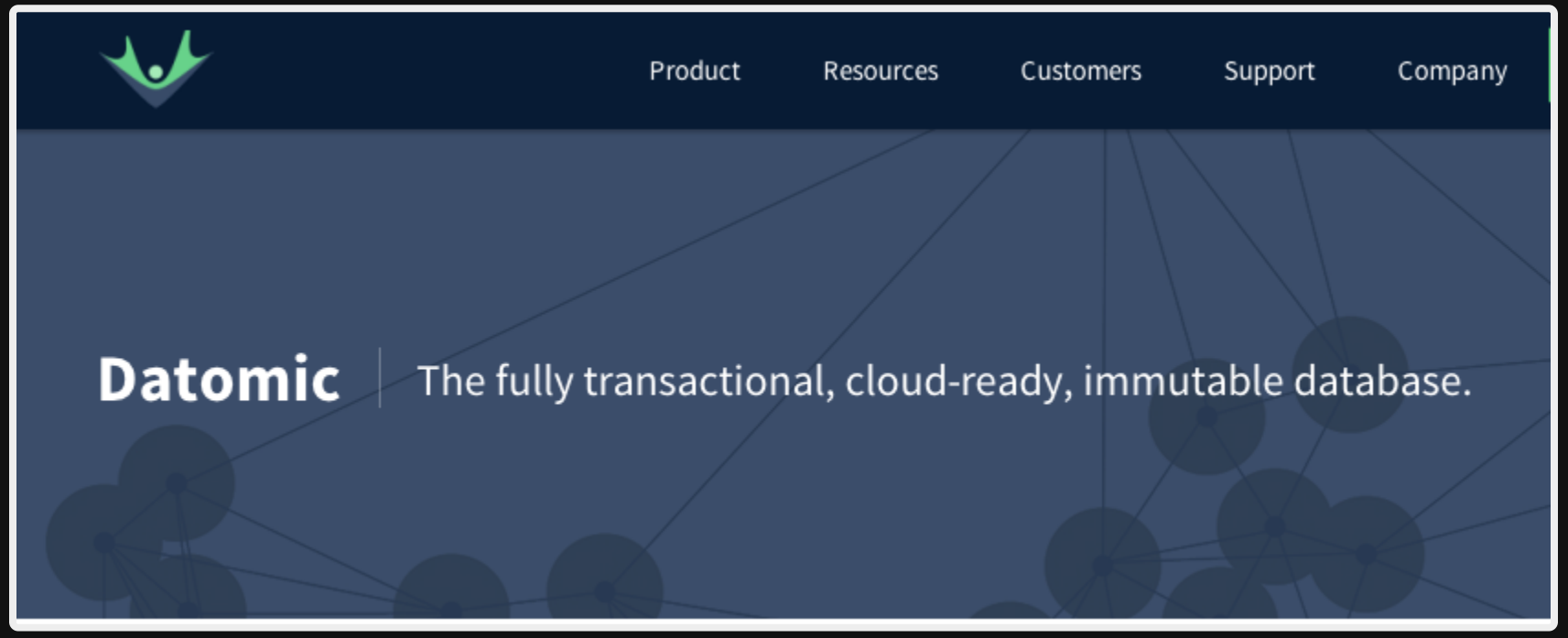
```clojure
;; Clojure

(def fields
  [:sighting/guid                    ;; "GLOBAL UNIQUE IDENTIFIER"
   :taxon/order                      ;; "TAXONOMIC ORDER"
   nil                               ;; "CATEGORY"
   :taxon/common-name                ;; "COMMON NAME"
   :taxon/scientific-name            ;; "SCIENTIFIC NAME"
   :taxon/subspecies-common-name     ;; "SUBSPECIES COMMON NAME"
   :taxon/subspecies-scientific-name ;; "SUBSPECIES SCIENTIFIC NAME"
   :sighting/count                   ;; "OBSERVATION COUNT" ;; x indicates uncounted
   ;; ...
   ])
```

Parsing the Data

```clojure
;; Clojure

(defn sighting-seq
  "Return a lazy sequence of lines from filename, transformed into sighting maps"
  [filename skip-rows nth-row]
  (->> (io/reader filename)
       (line-seq)
       (drop (or skip-rows 1))
       (take-nth nth-row)
       (map sighting)))
```

Data Storage and Query

Datomic



Datomic

- Schema
- Transactional
- History preserving
- Query language *is* Clojure data
- Results are Clojure data structures
- Query is executed in application server

Datomic

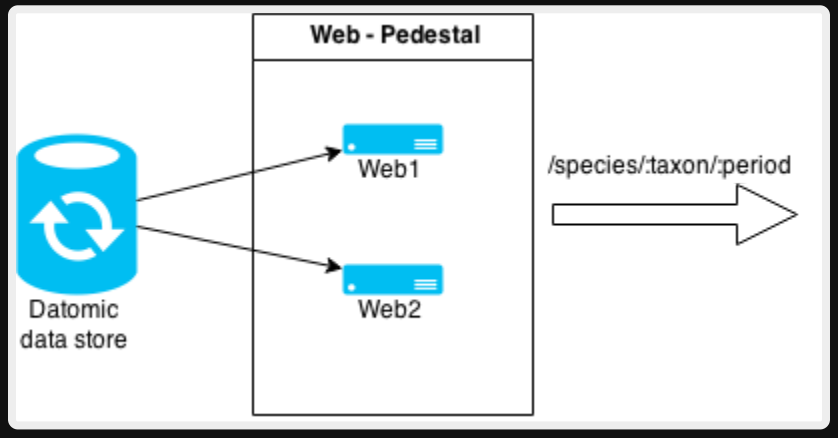
```
;; Clojure

(q '[:find (sum ?count) (count ?e)
     :where
     [?t :taxon/order "2881"]
     [?e :sighting/taxon ?t]
     [?e :sighting/state "Ohio"]
     [?e :sighting/county "Sandusky"]
     [?e :sighting/count ?count]]
   (db conn))

;;=> [[540 108]]
```

Web Service

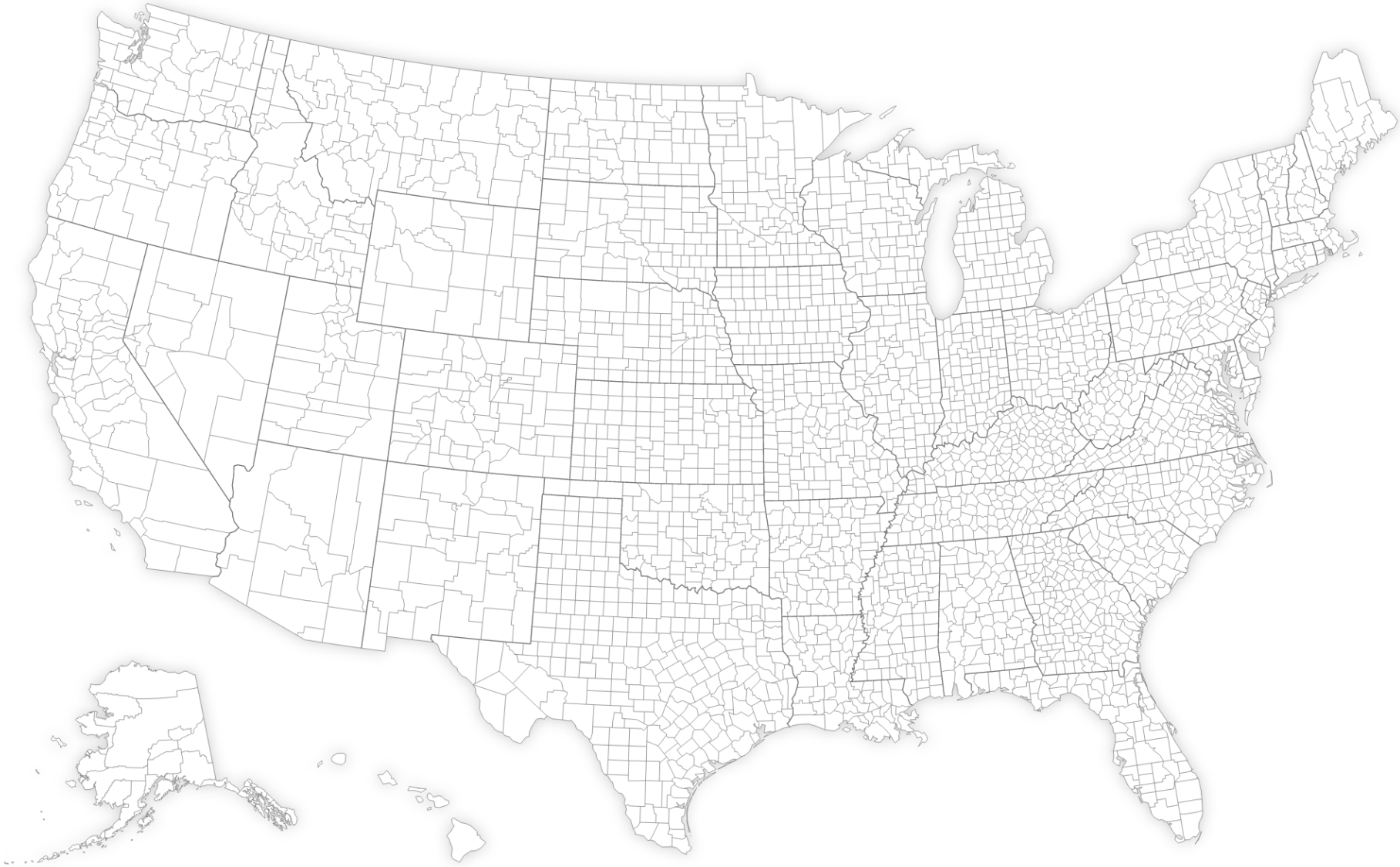
Pedestal

- Powerful middleware system
- Routing
- HTML Templating

Progress


Web - Pedestal
Web1
Web2
Datomic
data store
/species/:taxon/:period

Displaying the Data



Displaying the Data

Got ClojureScript?

Immediate wins:

- Easy to integrate into the existing stack
- Same language on both client and server
- Interoperability with JavaScript

What is ClojureScript?

Well, what is it, precious?

What is ClojureScript?

- Compiler for Clojure to JavaScript
- Emits JS optimized for the Google Closure library
- Several benefits over vanilla JS
  - Persistent data structures
  - Object keys as opposed to only strings
  - Laziness
  - Macros
  - Function argument destructuring

ClojureScript and JavaScript Interop

Methods

```
// JavaScript

var activeState = function() {
  return d3.select(".active");
}
```

```
;; ClojureScript

(defn active-state [] (.select js/d3 ".active"))
```

ClojureScript and JavaScript Interop

Properties

```
// JavaScript

var target = function() {
  return d3.event.target;
}
```

```
;; ClojureScript

(defn target [] (.-target (.-event js/d3)))
;; OR
(defn target [] (.. js/d3 -event -target)
```

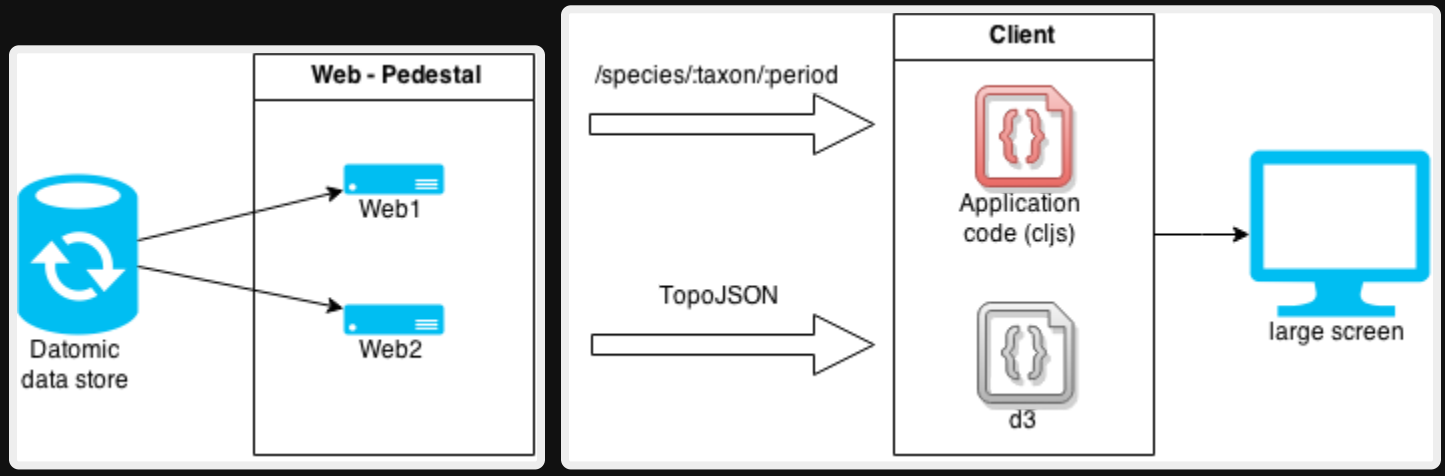
ClojureScript and JavaScript Interop

Fluent APIs and the –> Macro

```
// JavaScript

var months = d3.time.scale
               .domain([new Date(2012,10,15), new Date(2013,10,15)])
               .range([0, 900])
```

```
;; ClojureScript

(def months ( -> (js/d3.time.scale)
   (.domain (array (js/Date. 2012 10 15) (js/Date. 2013 10 15)))
   (.range (array 0 900))))
```

Progress


Web - Pedestal
Web1
Web2
Datomic
data store
Client
/species/:taxon/:period
TopoJSON
Application
code (cljs)
d3
large screen

Progress

Client-side Components

- Map
- Date slider
- Species list

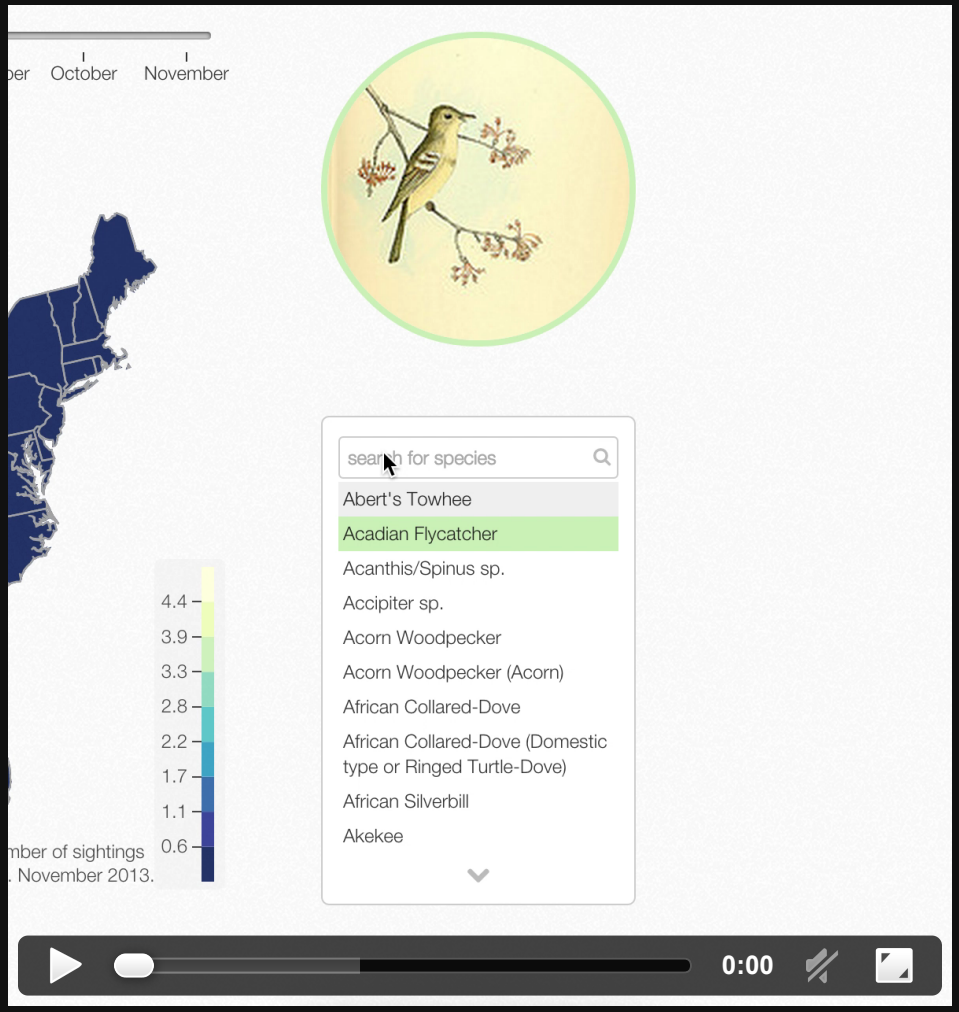
We Have a Prototype! But...

- Unpolished UI
- No structure to the data
- The database query was slow
- Differing views of end result
- Not responsive

React and Om

Imagination Land

An Om Component



Initial Render

Remember: Imagination Land

```
;; ClojureScript

(ul {:className "species-list"}
  (li {:className "species"} (a {:href "#/taxon/1"} "Abert's Towhee"))
  (li {:className "species"} (a {:href "#/taxon/2"} "Acadian Flycatcher"))
  (li {:className "species"} (a {:href "#/taxon/3"} "Acorn Woodpecker"))
  ... )
```

Building the List

```
;; ClojureScript

(defn species-li [species]
  (li {:className "species"}
      (a {:href (path species)} (:common-name species))))

(map species-li all-species)
```

Interactivity

```
;; ClojureScript

(map species-li
     (filter (match-string "black-thr") species))
```

Putting It Together

```
;; ClojureScript

(defn filter-list-items [filter-text species]
  (ul {:className "species-list"}
      (map species-li
           (filter (match-string filter-text) species))))
```

Browserland

We Don't Live in Imagination Land

React: A Better DOM

- Render the dom you want
- React takes care of the details
- Keeps real DOM in sync with your ideal DOM

Om

- Builds upon react
- Leverages ClojureScript's immutability

Om Example

```
;; ClojureScript

(defn species-list [model owner]
  (reify
    om/IRenderState
    (render-state [this state]
      (apply dom/ul #js {:className "species-list"}
             (om/build-all species-item
                           (:filtered-list state)
                           {:state (select-keys state [:highlighted :selected :select-c
```

Handling Events In Om

```clojure
;; ClojureScript

(defn species-item [model owner]
  (reify
    om/IRenderState
    (render-state [_ {:keys [highlighted selected select-ch]}]
      (dom/li #js {:className (str "taxon"
                                   (if (= model highlighted) " highlighted")
                                   (if (= model selected) " active"))}
        (dom/a #js {:href (taxon-path (:taxon/order model))
                    :onClick (fn [e]
                               (.preventDefault e)
                               (put! select-ch model))}
               (display-name model))))))
```

Brainsplosion

Integrating With Flickr

Integrating With Flickr

```
;; ClojureScript

(ns bird-wave.flickr (:require [cemerick.url :refer (url)]))

(def api-base-url (url "https://api.flickr.com/services/rest/"))
;;#cemerick.url.URL{:protocol "https", :username nil, :password nil, :host "api.flickr.

(str (assoc api-base-url :query {:api_key "my_super_flickr_key"}))
;; https://api.flickr.com/services/rest/?api_key=my_super_flickr_key
```

Integrating With Flickr

Updating the Model

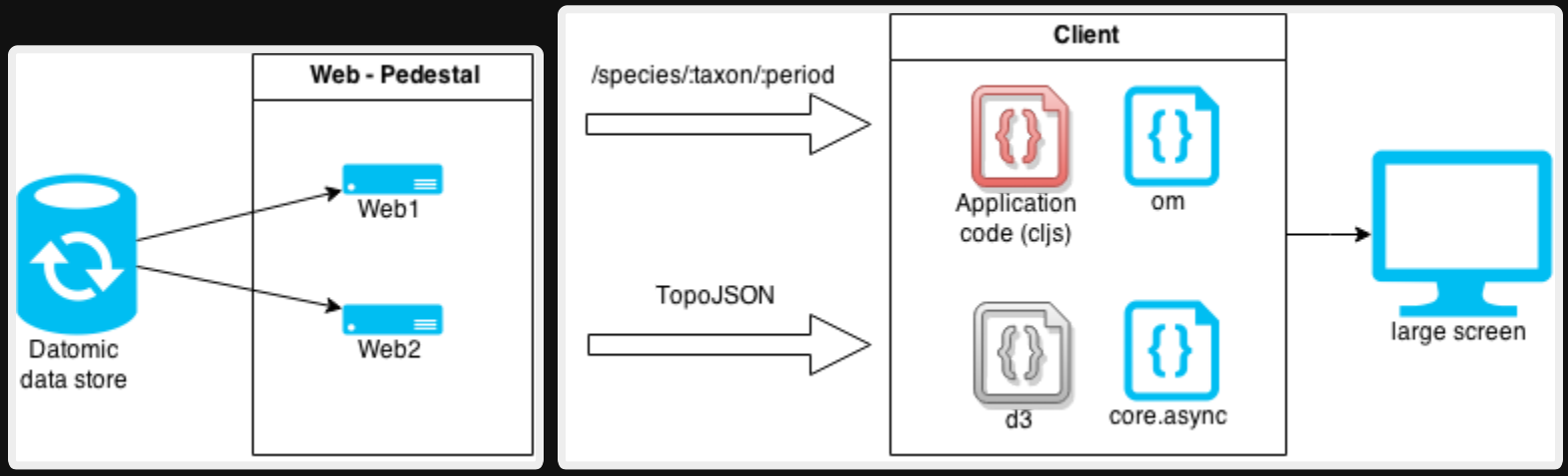
```
;; ClojureScript

(js/d3.json search-url (fn [data]
  (om/update! model :photo (first-photo data))))
;; { "photos": { "page": 1, "pages": "223", "perpage": 1, "total": "223",
;;  "photo": [
;;    { "id": "4769690133", "owner": "31064702@N05", "secret": "818406d0cd", "server":
;;  ] }, "stat": "ok" }

(js/d3.json url (fn [data]
  (om/update! model :attribution (attribution data))))
;; { "photo": { "id": "4769690133", "secret": "818406d0cd", "server": "4123", "farm": 5
;; "owner": { "nsid": "31064702@N05", "username": "Dawn Huczek", "realname": "", "locat
;; ...
;; "urls": {
;;   "url": [
;;     { "type": "photopage", "_content": "https:\/\/www.flickr.com\/photos\/31064702@N
;;   ] }, "media": "photo" }, "stat": "ok" }
```

Integrating With Flickr

The Selection Image Component

```
;; ClojureScript
(dom/div #js {:id "selection-image"
              :className (if (seq model) "loaded" "no-photo")}
  (dom/img #js {:className "photo"
                :src (try-with-default model :url_q "/images/loading.png")})
  (dom/div #js {:className "attribution"}
    (dom/h3 #js {:className "title"}
      (try-with-default model :title "No photo available"))
    (dom/div #js {:className "by"}
      …
      (if (seq (:attribution model))
        (dom/a #js {:className "detail fetched"
                    :href (get-in model [:attribution :url])
                    :target "_blank"}
               (get-in model [:attribution :by]))
        (dom/a #js {:className "detail"
                    :href "#"
                    :onClick #(fetch-attribution % model)}
               "view attribution")))))
```

Progress

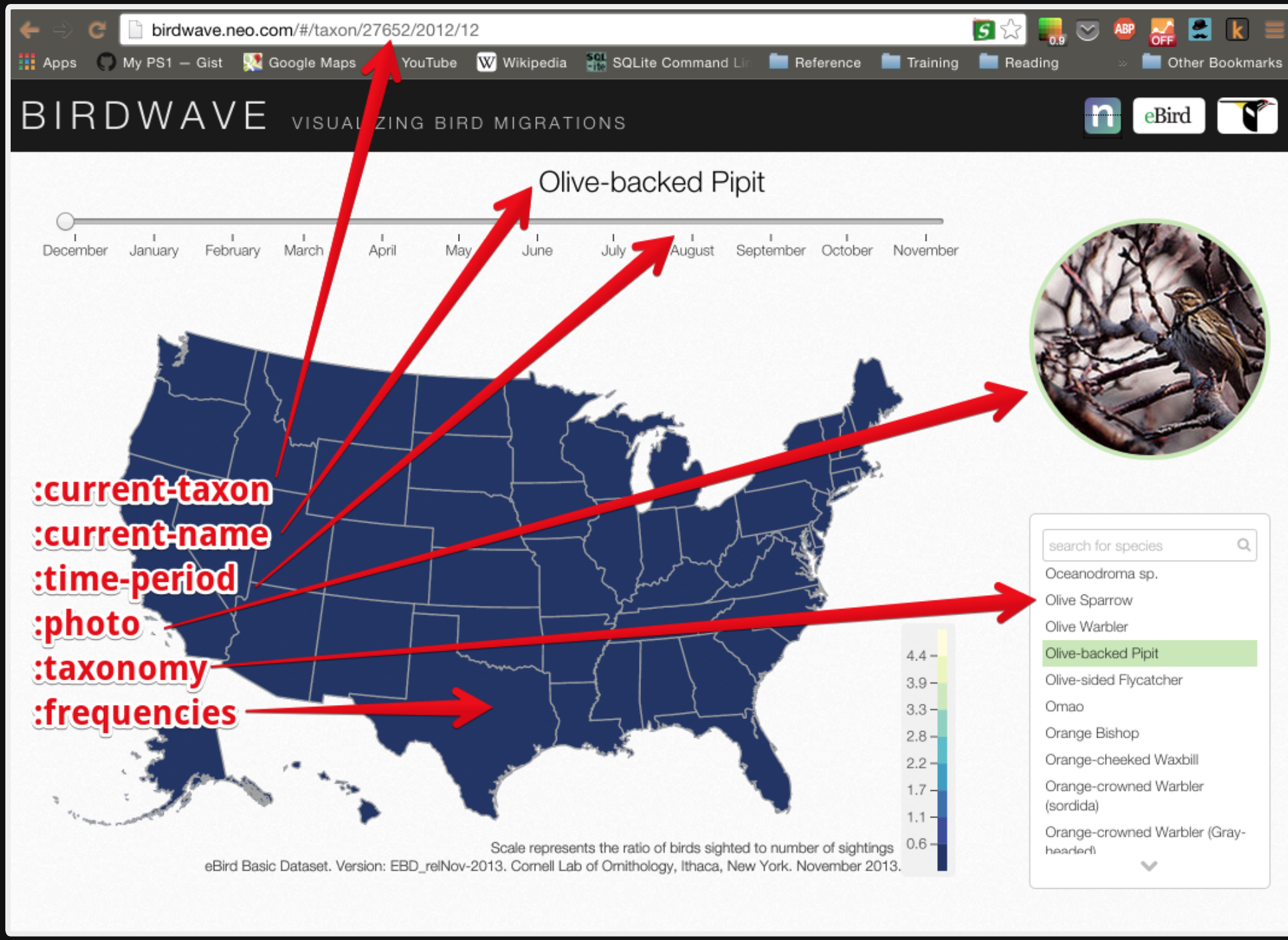

Web - Pedestal
Web1
Web2
Datomic
data store
/species/:taxon/:period
TopoJSON
Client
Application
code (cljs)
om
d3
core.async
large screen

Progress

Client-side Components

- Map
- Date slider
- Species list
- Push state
- Bird photo
- Header
- Typeahead

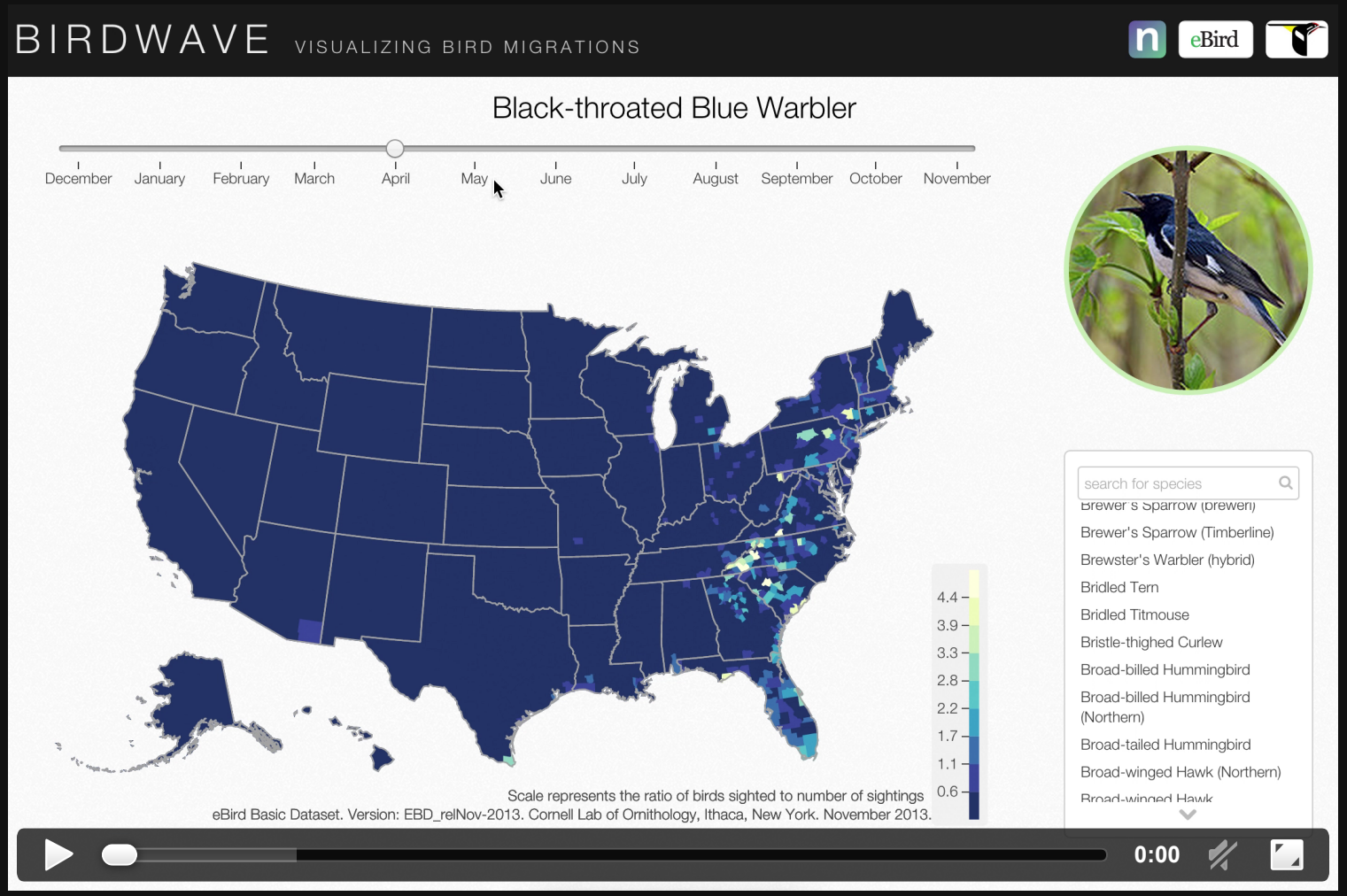
Birdwave with Om


birdwave.neo.com/#/taxon/27652/2012/12
Apps
My PS1 — Gist
Google Maps
YouTube
Wikipedia
SQLite Command Lir
Reference
Training
Reading
Other Bookmarks
BIRDWAVE
VISUALIZING BIRD MIGRATIONS
eBird
Olive-backed Pipit
December
January
February
March
April
May
June
July
August
September
October
November
:current-taxon
:current-name
:time-period
:photo
:taxonomy
:frequencies
4.4
3.9
3.3
2.8
2.2
1.7
1.1
0.6
Scale represents the ratio of birds sighted to number of sightings
eBird Basic Dataset. Version: EBD_relNov-2013. Cornell Lab of Ornithology, Ithaca, New York. November 2013.
search for species
Oceanodroma sp.
Olive Sparrow
Olive Warbler
Olive-backed Pipit
Olive-sided Flycatcher
Omao
Orange Bishop
Orange-cheeked Waxbill
Orange-crowned Warbler (sordida)
Orange-crowned Warbler (Gray-headed)

This Is Way Better! But...

- Not responsive

Adding Responsiveness



Adding Responsiveness

- CSS is not enough
- Screen size *is* user input

Adding Responsiveness

Client changes

- Detect and store screen size in app state
  - xs: 0px < width <= 520px
  - sm: 520px < width <= 768px
  - md: 768px < width <= 1024px
  - lg: 1024px < width

Adding Responsiveness

```
;; ClojureScript

(defn watch-screen-size [model]
  (let [size-handler (fn [size]  #(swap! model assoc :screen-size size))]
    (-> js/enquire
      (.register "screen and (min-width: 0px) and (max-width: 520px)"    (size-handler
      (.register "screen and (min-width: 521px) and (max-width: 768px)"  (size-handler
      (.register "screen and (min-width: 769px) and (max-width: 1024px)" (size-handler
      (.register "screen and (min-width: 1025px)"                        (size-handler
```

Adding Responsiveness

Client changes

- Update necessary components to render accordingly
  - Map renders states vs counties on md, sm, xs
  - Photo does not render on md, sm, xs
  - Slider changes to select widget on sm, xs
- Update ajax calls
  - Map data
  - API requests

Adding Responsiveness

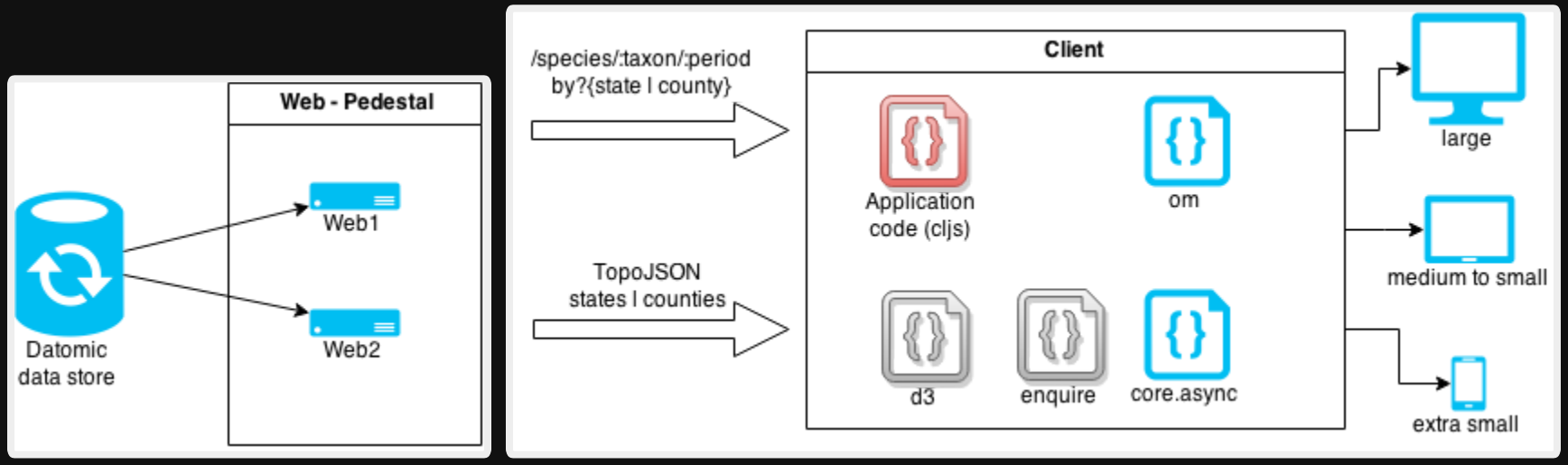
```
;; ClojureScript

;; display the slider on large screens, and the select widget on small
(if (contains? #{"lg" "md"} (:screen-size model))
  (om/build date-slider model {:state {:time-period-ch time-period-ch}})
  (om/build date-select model {:state {:time-period-ch time-period-ch}}))
```

Adding Responsiveness

Server changes

- Handle query parameters on API requests
- Add queries to return data aggregated by state vs county

Progress


Web - Pedestal
Web1
Web2
Datomic
data store
/species/:taxon/:period
by?{state | county}
TopoJSON
states | counties
Client
Application
code (cljs)
om
d3
enquire
core.async
large
medium to small
extra small

Ship It!

Resources

Birdwave

- Announcing Birdwave
- Choropleths and d3js
- Using the Flickr API with ClojureScript
- Adding an Om Component -- a Walkthrough
- Responsive JavaScript with EnquireJS

Resources

Birding

- eBird.org
- Cornell Lab of Ornithology
- Birding portal by Cornell

Resources

Clojure

- clojure.org
- Are We There Yet? - Rich Hickey
- Datomic
- Pedestal
- core.async

Resources

ClojureScript

- The ClojureScript Compilation Pipeline
- David Nolen on ClojureScript and Om
- David Nolen's seminal post on Om's approach
- Running ClojureScript with Node
- React.js
- Be Predictable, Not Correct (React.js)

Resources

d3

- d3js Home page
- Scott Murray's tutorials...
- ... and his book
- Mike Bostock's simple map example
- Building a Choropleth
- Play with GeoJSON