

Visual Interfaces in ClojureScript

Kevin Lynagh

Keming Labs @lynaghk

2012 September

Strange Loop
St. Louis, MO USA

Agenda

Agenda



why

Visual Interfaces

& *a helpful way
to think about 'em*

Agenda

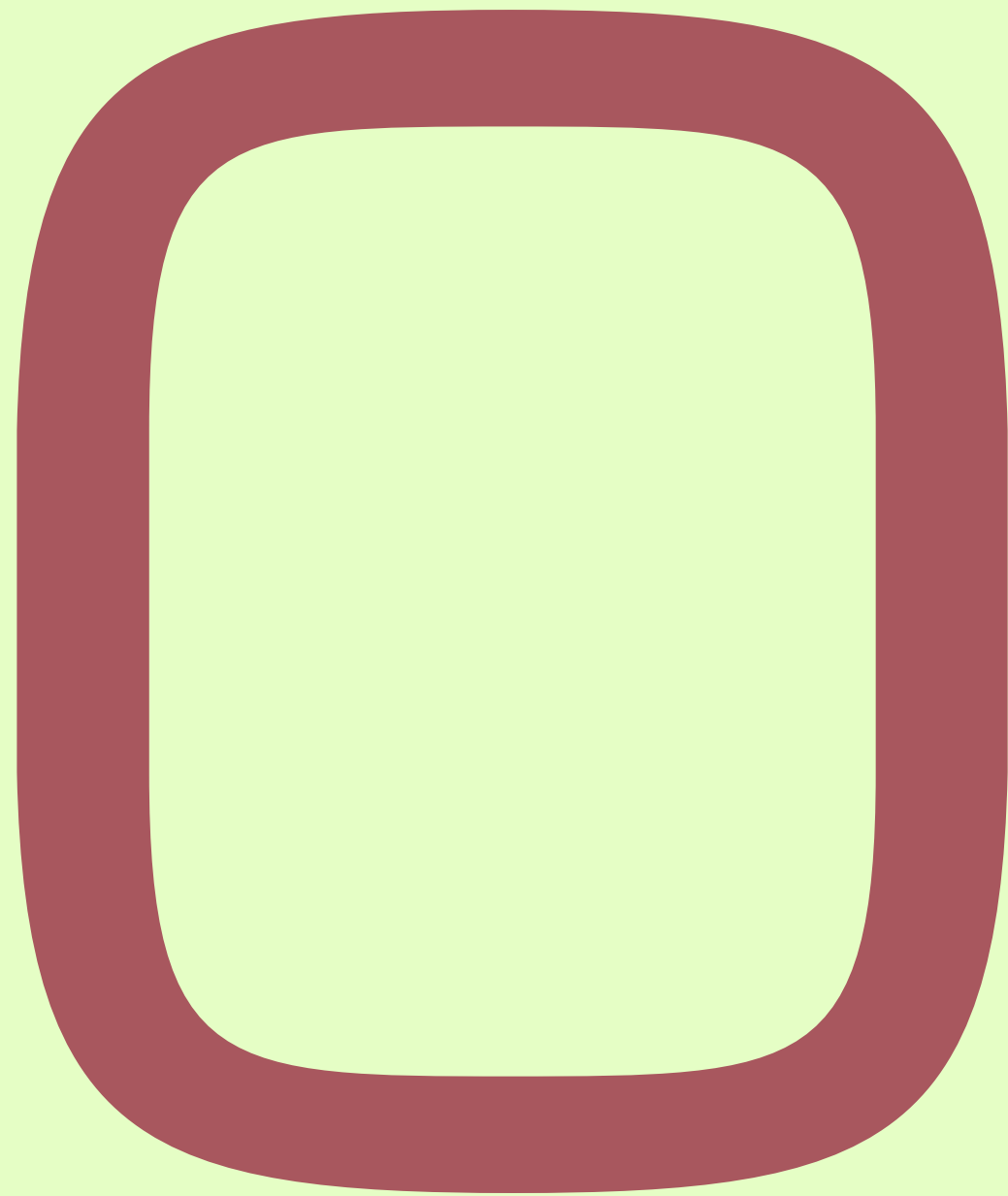
an

Example

Agenda

G Some *Implications*

Anti-Agenda



~~Tech~~

ideasTalk

Agenda



why

Visual Interfaces

& *a helpful way
to think about 'em*

Bioinformatics



Chrome File Edit View History Bookmarks Window Help

localhost:8080

localhost:8080

VCF Visualizer

Logged in as: keminglabs

VCF files

Select VCF files

Export subset

Wind energy



What
do these things
have in
common



Data



Clojure



Philosophy

State :: GOTO

My second remark is that our intellectual powers are rather geared to master static relations and that **our powers to visualize processes evolving in time are relatively poorly developed**. For that reason we should do (as wise programmers aware of our limitations) our utmost to shorten the conceptual gap between the static program and the dynamic process, to make the correspondence between the program (spread out in text space) and the process (spread out in time) as trivial as possible.

Edsger Dijkstra

Go To Statement Considered Harmful

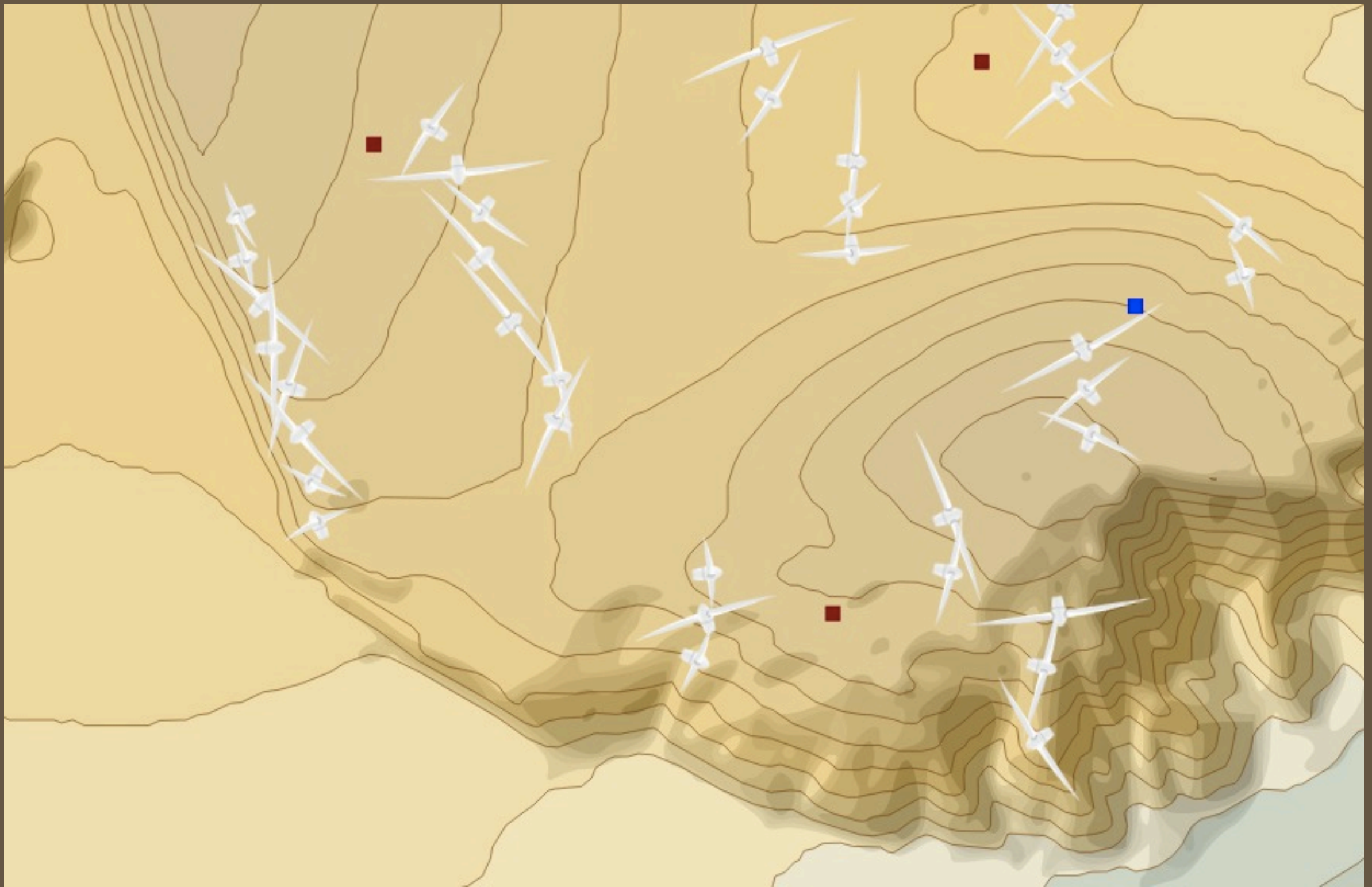
Program Process

- I. Explicit mutable state
- II. Explicit state → visual mapping
- III. Let the computer do the work

Agenda

an

Example



What is the essential state?

```
(def !turbines (atom [  
  { :id "840009029", :latitude 1234, :longitude 5678  
    :wind-speed 26.04, :wind-direction 5.49}  
  
  { :id "840009030", :latitude 1234, :longitude 5678  
    :wind-speed 43.95, :wind-direction 3.86}  
  
  ... ]))
```

Data



Data



Treat your *views* like Data

```
[ :div.turbine-spinner  
  { :class :speed-2  
    :style { :left "12.3%", :top "78.4%"  
      :-webkit-transform  
        "rotate3d(0,0,1,5.49rad)" } } ]
```


Templates = Functions

```
(defn spinner [turbine]
  (let [{:keys [latitude longitude
                wind-speed wind-direction]} turbine]

    [:div.turbine-spinner
     {:class (speed-category wind-speed)
      :style {:left (str (scale-x longitude) "%")
               :top (str (scale-y latitude) "%")
               :-webkit-transform
                (str "rotate3d(0,0,1," wind-direction "rad)"} )}}]))
```

Templates = Functions

```
(defn spinner [turbine]
  (let [{:keys [latitude longitude
                wind-speed wind-direction]} turbine]

    [:div.turbine-spinner
     {:class (speed-category wind-speed)
      :style {:left (str (scale-x longitude) "%")
               :top (str (scale-y latitude) "%")
               :-webkit-transform
                (str "rotate3d(0,0,1," wind-direction "rad)")} } ]))
```

Templates = Functions

```
(defn spinner [turbine]
  (let [{:keys [latitude longitude
                wind-speed wind-direction]} turbine]

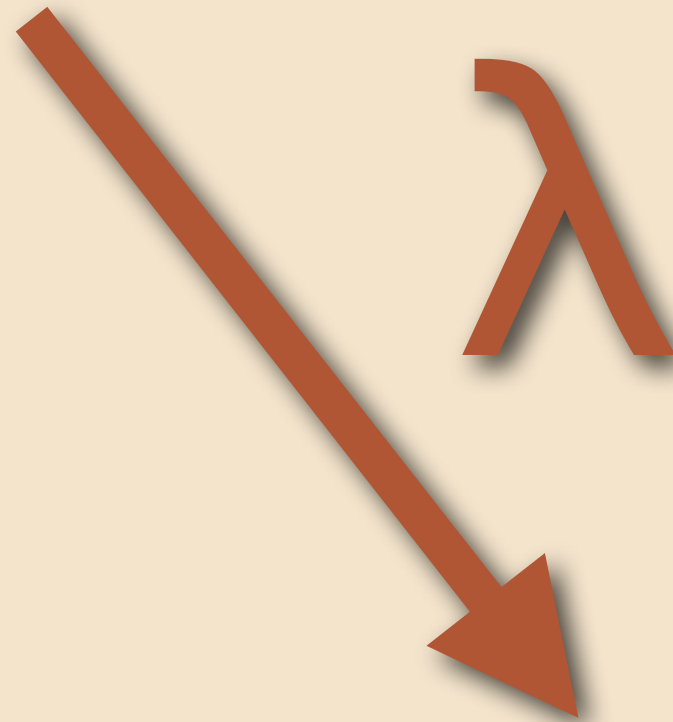
    [:div.turbine-spinner
     {:class (speed-category wind-speed)
      :style (merge (position turbine)
                    {:-webkit-transform
                     (str "rotate3d(0,0,1," wind-direction "rad)"))})])])

(defn position [{:keys [latitude longitude]}]
  {:left (str (scale-x longitude) "%")
   :top  (str (scale-y latitude) "%")})
```



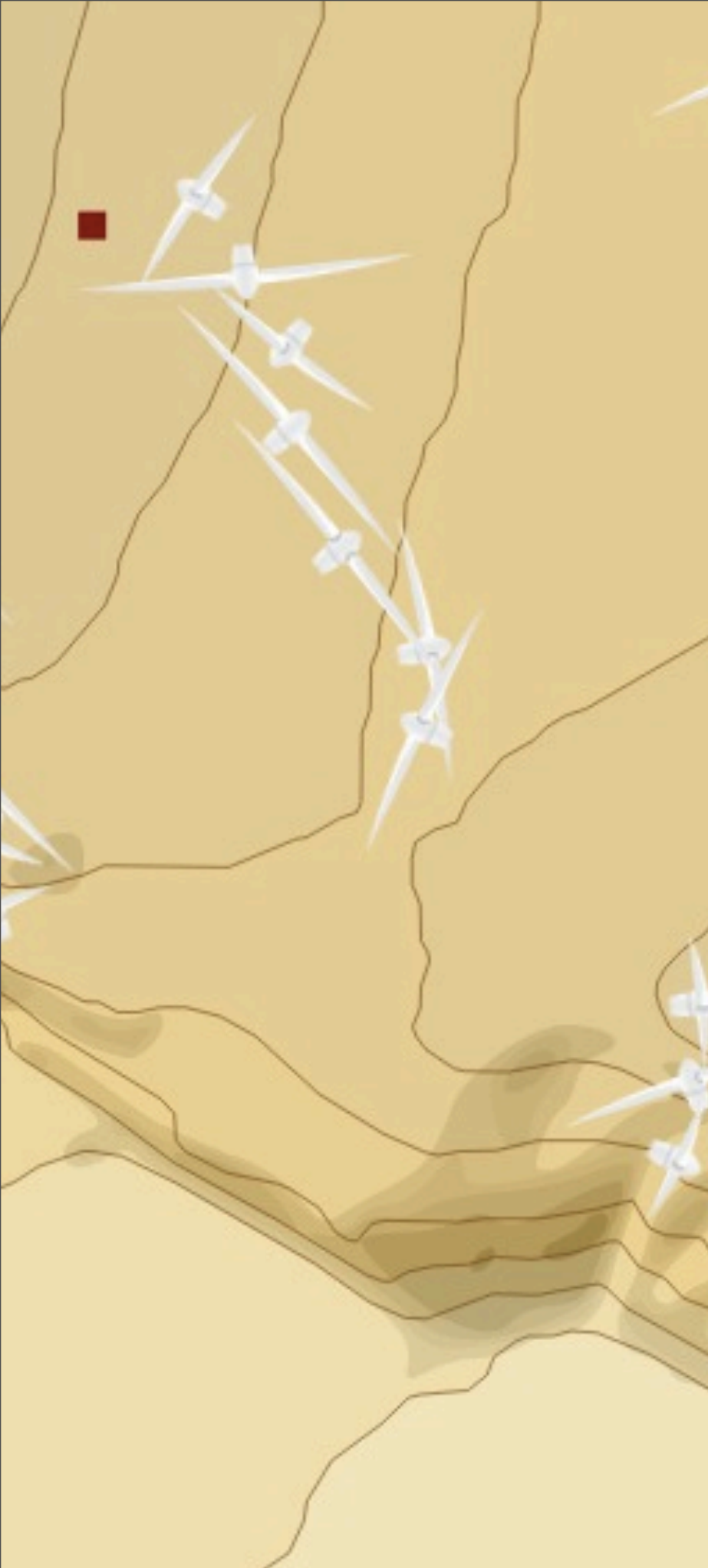
```
{:id "840009029", :latitude 1234,  
:longitude 5678, :wind-speed 26.04  
:wind-direction 5.49}
```

Domain data

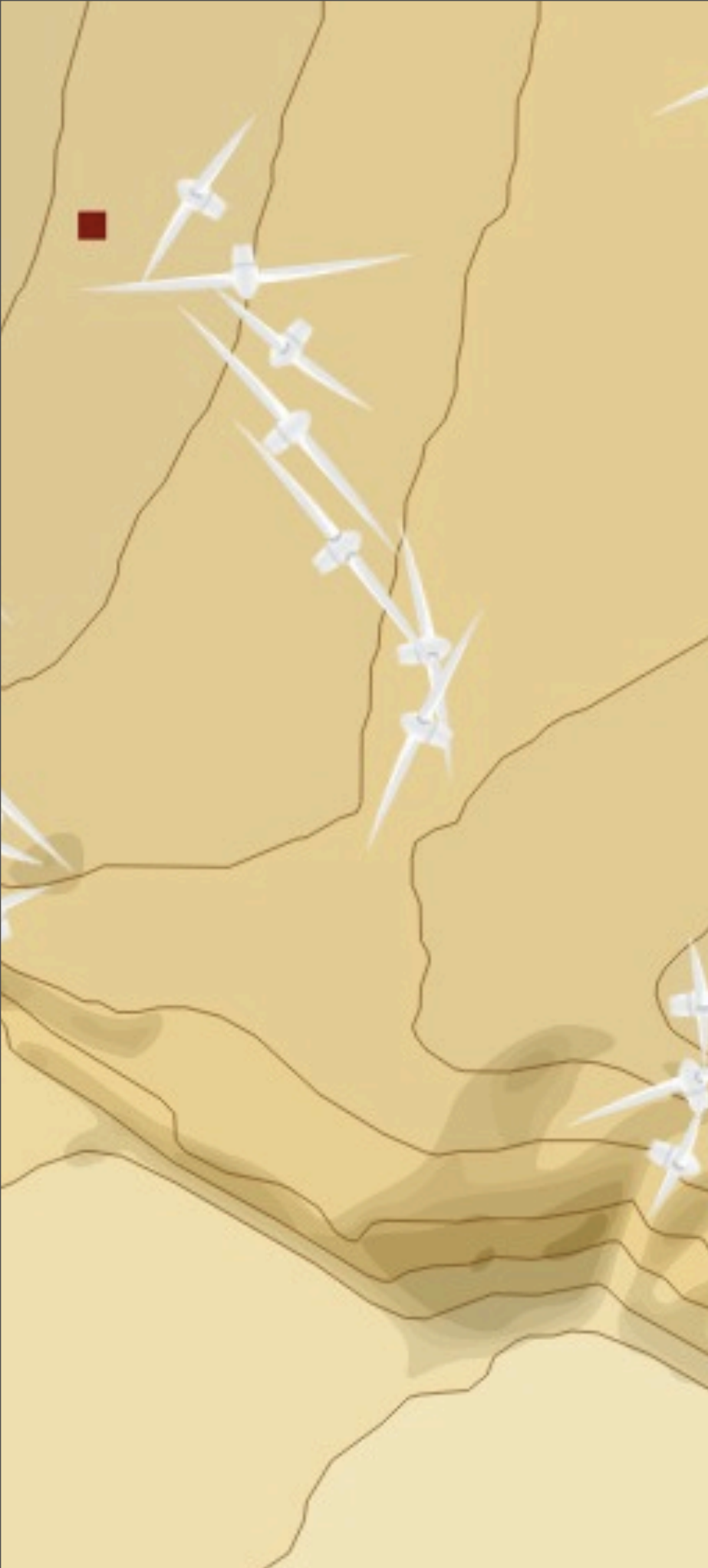


Markup
(still data)

```
[ :div.turbine-spinner  
  { :class :speed-2  
    :style { :left "12.3%", :top "78.4%"  
             :-webkit-transform  
              "rotate3d(0,0,1,5.49rad)" } } ]
```



```
( render!  
  [ :div#turbine-spinners  
    ( map spinner @!turbines ) ] )
```



(**render!**

```
[ :div#turbine-spinners
```

```
  [:div.turbine-spinner
```

```
    {:class :speed-2
```

```
    :style {:left "12.3%", :top "78.4%"
```

```
      :-webkit-transform
```

```
        "rotate3d(0,0,1,5.49rad)"}}]
```

```
  [:div.turbine-spinner
```

```
    {:class :speed-2
```

```
    :style {:left "12.3%", :top "78.4%"
```

```
      :-webkit-transform
```

```
        "rotate3d(0,0,1,5.49rad)"}}]
```

```
  [:div.turbine-spinner
```

```
    {:class :speed-2
```

```
    :style {:left "12.3%", :top "78.4%"
```

```
      :-webkit-transform
```

```
        "rotate3d(0,0,1,5.49rad)"}}]
```

```
  [:div.turbine-spinner
```

```
    {:class :speed-2
```

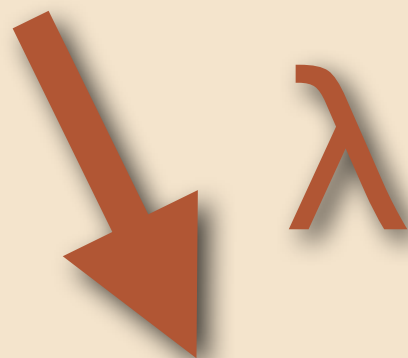
```
    :style {:left "12.3%", :top "78.4%"
```

```
      :-webkit-transform
```

```
        "rotate3d(0,0,1,5.49rad)"}}]
```

```
... ])
```

Domain data

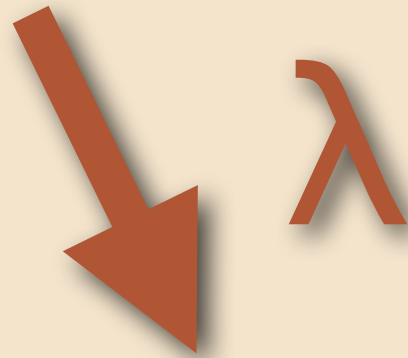


Markup

render!



Domain data

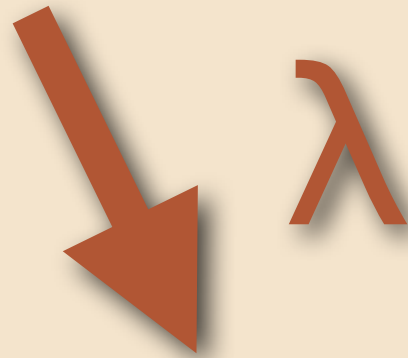


Markup

render!



New
Domain data



Markup

merge!





[:div#turbine-spinners

```
[ :div.turbine-spinner  
  { :class :speed-2  
    :style { :left "12.3%", :top "78.4%"  
             :-webkit-transform  
               "rotate3d(0,0,1,5.49rad)" } } ]
```

```
[ :div.turbine-spinner  
  { :class :speed-2  
    :style { :left "12.3%", :top "78.4%"  
             :-webkit-transform  
               "rotate3d(0,0,1,5.49rad)" } } ]
```

...]



```
<!DOCTYPE html>  
▼ <html lang="en">  
  ► <head>...</head>  
  ▼ <body>  
    ▼ <div id="map">  
      ► <svg preserveAspectRatio="xMidYMax" viewBox="-1  
4016285.587683 5009.377086 4993.822085">...</svg>  
      ▼ <div id="turbine-spinners">  
        ► <div style="left: 56.738308983244565%; top: 4  
class="speed-1">...</div>  
        ► <div style="left: 57.00804221286055%; top: 50  
class="speed-2">...</div>  
        ► <div style="left: 57.119670890966646%; top: 5  
class="speed-1">...</div>  
        ► <div style="left: 55.01468179855416%; top: 62  
class="speed-4">...</div>
```



[:div#turbine-spinners

```
[ :div.turbine-spinner  
  { :class :speed-2  
    :style { :left "12.3%", :top "78.4%"  
      :-webkit-transform  
        "rotate3d(0,0,1,5.49rad)" } } ]
```

```
[ :div.turbine-spinner  
  { :class :speed-2  
    :style { :left "12.3%", :top "78.4%"  
      :-webkit-transform  
        "rotate3d(0,0,1,5.49rad)" } } ]
```

...]



```
<!DOCTYPE html>  
▼ <html lang="en">  
  ▶ <head>...</head>  
  ▼ <body>  
    ▼ <div id="map">  
      ▶ <svg preserveAspectRatio="xMidYMax" viewBox="-1  
4016285.587683 5009.377086 4993.822085">...</svg>  
      ▼ <div id="turbine-spinners">  
        ▶ <div style="left: 56.738308983244565%; top: 4  
class="speed-1">...</div>  
        ▶ <div style="left: 57.00804221286055%; top: 50  
class="speed-2">...</div>  
        ▶ <div style="left: 57.119670890966646%; top: 5  
class="speed-1">...</div>  
        ▶ <div style="left: 55.01468179855416%; top: 62  
class="speed-4">...</div>
```




[:div#turbine-spinners

```
[ :div.turbine-spinner  
  { :class :speed-2  
    :style { :left "12.3%", :top "78.4%"  
             :-webkit-transform  
               "rotate3d(0,0,1,5.49rad)" } } ]
```

```
[ :div.turbine-spinner  
  { :class :speed-2  
    :style { :left "12.3%", :top "78.4%"  
             :-webkit-transform  
               "rotate3d(0,0,1,5.49rad)" } } ]
```

...]



```
<!DOCTYPE html>  
▼ <html lang="en">  
  ▶ <head>...</head>  
  ▼ <body>  
    ▼ <div id="map">  
      ▶ <svg preserveAspectRatio="xMidYMax" viewBox="-1  
4016285.587683 5009.377086 4993.822085">...</svg>  
      ▼ <div id="turbine-spinners">  
        ▶ <div style="left: 56.738308983244565%; top: 4  
class="speed-1">...</div>  
        ▶ <div style="left: 57.00804221286055%; top: 50  
class="speed-2">...</div>  
        ▶ <div style="left: 57.119670890966646%; top: 5  
class="speed-1">...</div>  
        ▶ <div style="left: 55.01468179855416%; top: 62  
class="speed-4">...</div>
```



[:div#turbine-spinners

```
[ :div.turbine-spinner  
  { :class :speed-2  
    :style { :left "12.3%", :top "78.4%"  
             :-webkit-transform  
               "rotate3d(0,0,1,5.49rad)" } } ]
```

```
[ :div.turbine-spinner  
  { :class :speed-2  
    :style { :left "12.3%", :top "78.4%"  
             :-webkit-transform  
               "rotate3d(0,0,1,5.49rad)" } } ]
```

...]



```
<!DOCTYPE html>  
▼ <html lang="en">  
  ▶ <head>...</head>  
  ▼ <body>  
    ▼ <div id="map">  
      ▶ <svg preserveAspectRatio="xMidYMax" viewBox="-1  
4016285.587683 5009.377086 4993.822085">...</svg>  
      ▼ <div id="turbine-spinners">  
        ▶ <div style="left: 56.738308983244565%; top: 4  
class="speed-1">...</div>  
        ▶ <div style="left: 57.00804221286055%; top: 50  
class="speed-2">...</div>  
        ▶ <div style="left: 57.119670890966646%; top: 5  
class="speed-1">...</div>  
        ▶ <div style="left: 55.01468179855416%; top: 62  
class="speed-4">...</div>
```

```
( bind!  "#turbine-spinners"  
  [ :div#turbine-spinners  
    ( map spinner @!turbines ) ] )
```

```
(bind! "#turbine-spinners"  
  [:div#turbine-spinners  
    (map spinner @!turbines)])
```




```
(bind! "#turbine-spinners"  
  [:div#turbine-spinners  
    (map spinner @!turbines)])
```



C2



<http://github.com/lynaghk/c2>

todos

What needs to be done?

Double-click to edit a todo

Template by [Sindre Sorhus](#)

Created by [Kevin Lynagh](#)

Part of [TodoMVC](#)

Necessary State

```
(def !todos  
  "Todo list, implicitly key'd by :title"  
  (atom []))
```

```
(def !filter  
  "Which todo items should be displayed:  
  all, active, or completed?"  
  (atom :all))
```


Template function

```
(defn todo* [t]
  (let [{:keys [completed? title editing?]} t]
    [:li {:class (str (when completed? "completed")
                      " " (when editing? "editing"))}]
      [:div.view
        [:input.toggle {:type "checkbox"
                        :properties {:checked completed?}}]
        [:label title]
        [:button.destroy]]
        [:input.edit {:value title}]]))
```

Declarative

```
(bind! "#main"  
  [:section#main  
    {:style {:display (when (zero? (count @core/!todos)) "none")}}}  
    [:input#toggle-all  
      {:type "checkbox"  
        :properties {:checked (every? :completed? @core/!todos)}}]  
    [:label {:for "toggle-all"} "Mark all as complete"]  
    [:ul#todo-list  
      (map todo* (case @core/!filter  
        :active      (remove :completed? @core/!todos)  
        :completed   (filter :completed? @core/!todos)  
        ;;default to showing all events  
        @core/!todos))]])
```

Declarative

```
(bind! "#main"
  [:section#main
    {:style {:display (when (zero? (count @core/!todos)) "none"))}}
  [:input#toggle-all
    {:type "checkbox"
     :properties {:checked (every? :completed? @core/!todos)}}]
  [:label {:for "toggle-all"} "Mark all as complete"]
  [:ul#todo-list
    (map todo* (case @core/!filter
                  :active      (remove :completed? @core/!todos)
                  :completed (filter :completed? @core/!todos)
                  ;;default to showing all events
                  @core/!todos)))]])
```

Declarative

```
(bind! "#main"
  [:section#main
    {:style {:display (when (zero? (count @core/!todos)) "none"))}}]
  [:input#toggle-all
    {:type "checkbox"
     :properties {:checked (every? :completed? @core/!todos)}}]
  [:label {:for "toggle-all"} "Mark all as complete"]
  [:ul#todo-list
    (map todo* (case @core/!filter
                  :active      (remove :completed? @core/!todos)
                  :completed (filter :completed? @core/!todos)
                  ;;default to showing all events
                  @core/!todos)))]])
```

Declarative

```
(bind! "#main"
  [:section#main
    {:style {:display (when (zero? (count @core/!todos)) "none"))}}
  [:input#toggle-all
    {:type "checkbox"
     :properties {:checked (every? :completed? @core/!todos)}}]
  [:label {:for "toggle-all"} "Mark all as complete"]
  [:ul#todo-list
    (map todo* (case @core/!filter
      :active      (remove :completed? @core/!todos)
      :completed   (filter :completed? @core/!todos)
      ;;default to showing all events
      @core/!todos)))]])
```

todos



What needs to be done?



~~Talk outline~~



Example application



Visual stuff



Q & A / Moar beer

3 items left

All Active Completed

Clear completed (1)

Double-click to edit a todo

Template by [Sindre Sorhus](#)

Created by [Kevin Lynagh](#)

todos

» *What needs to be done?*

✓ ~~Talk outline~~

✓ Example application

✓ Visual stuff



github.com/lynaghk/c2-demos

Template by [Sindre Sorhus](#)

Created by [Kevin Lynagh](#)

Agenda

G Some *Implications*

Program Process

- I. Explicit mutable state
- II. Explicit state → visual mapping
- III. Let the computer do the work

De

1) specification

2) rendering

coupled

Runtime flexibility

JVM JavaScript

CLR *Lua* C

Easy testing

(no browser needed)

Tradeoffs

Performance

DOM walking always slower than direct references; when does it matter?

Coupling markup/dev

Angry designers?

Less code

Write in forward direction

Minimal DOM references

Clear dataflow

Kevin Lynagh

Keming Labs @lynaghk

