BERLIN, 22 Feb 2018

JOY CLARK

Web Applications ❤️ Clojure

INNOQ

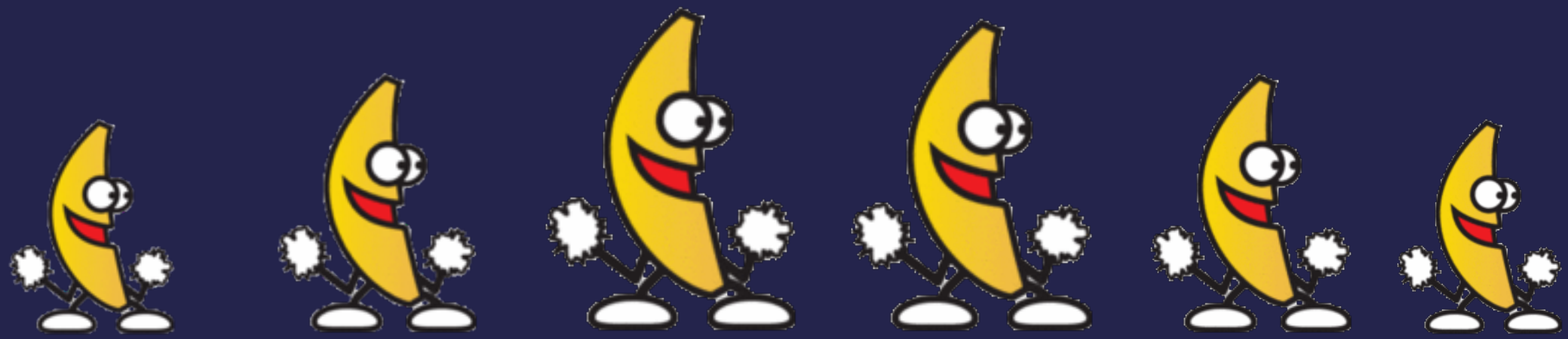
Joy Clark

Consultant @ innoQ

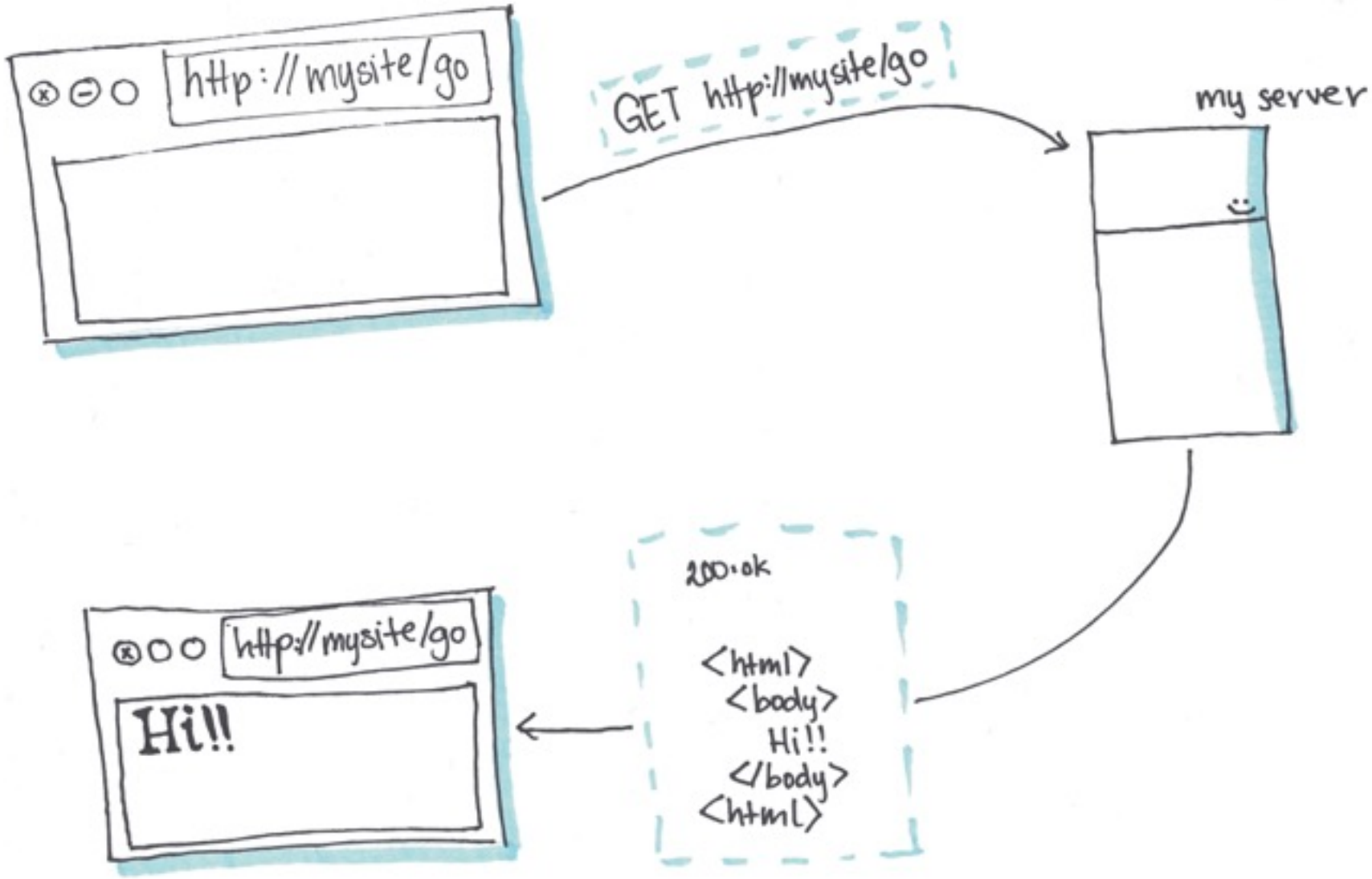
joy.clark@innoq.com

@iamjoyclark

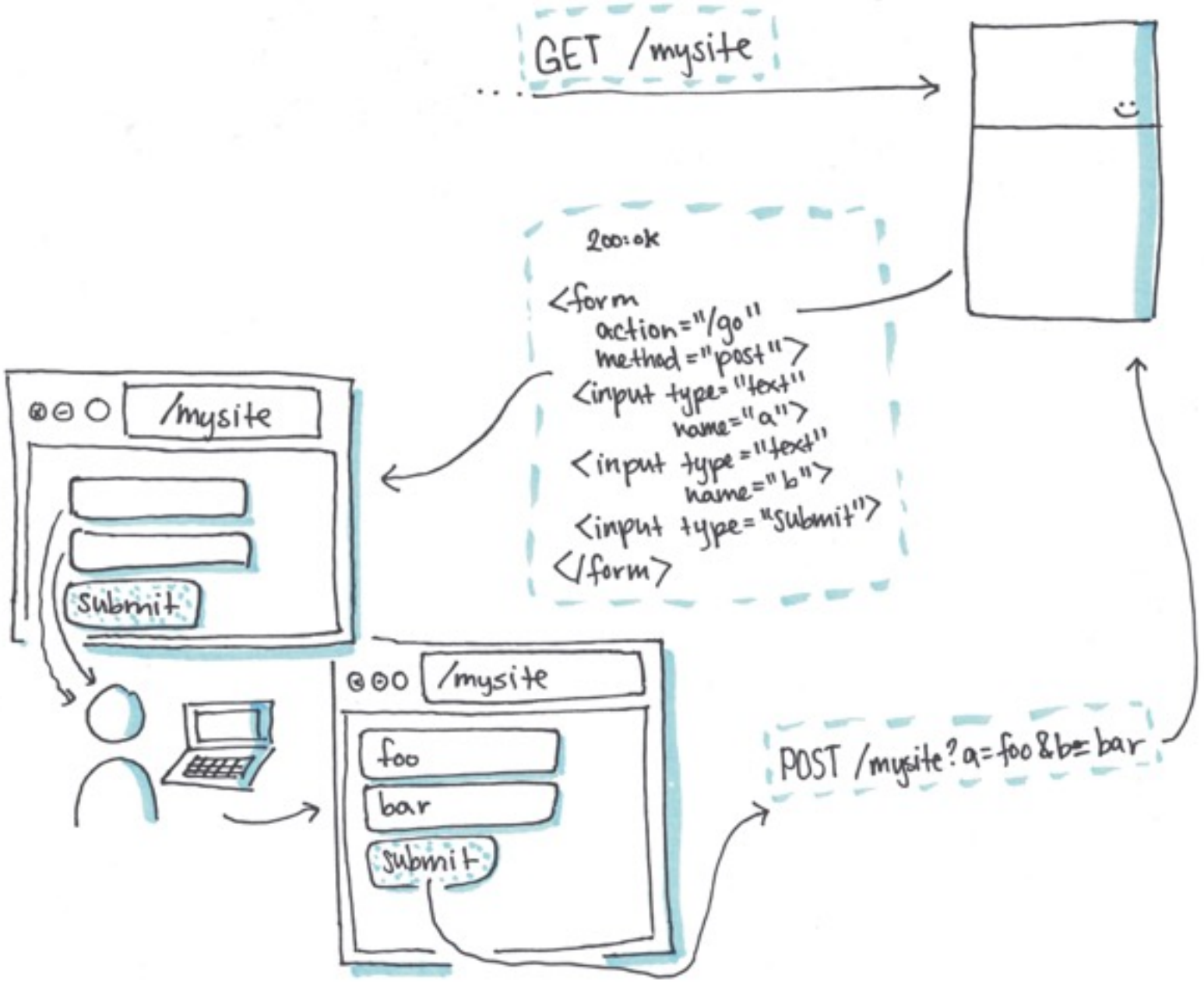
Web Applications



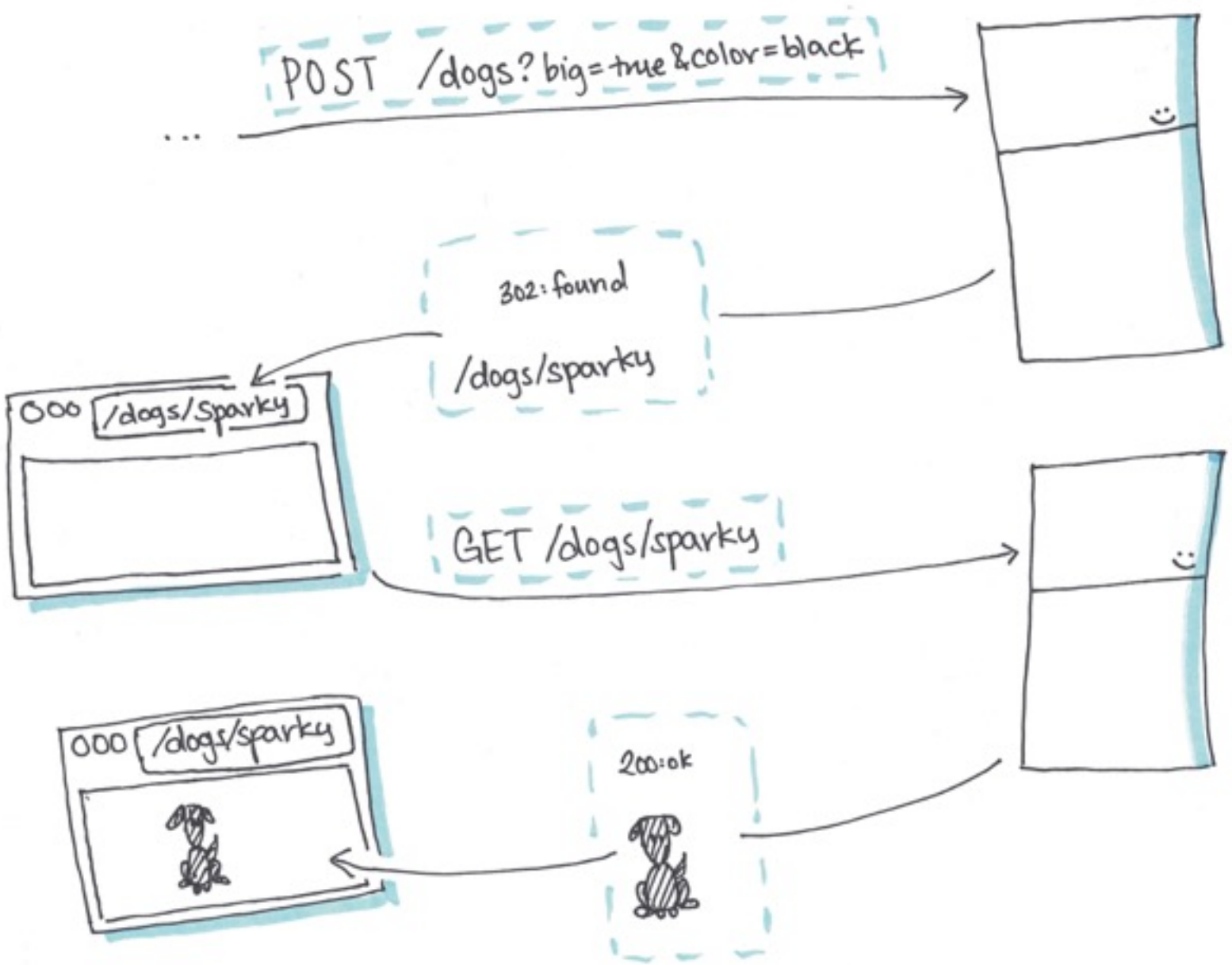
retrieving a page



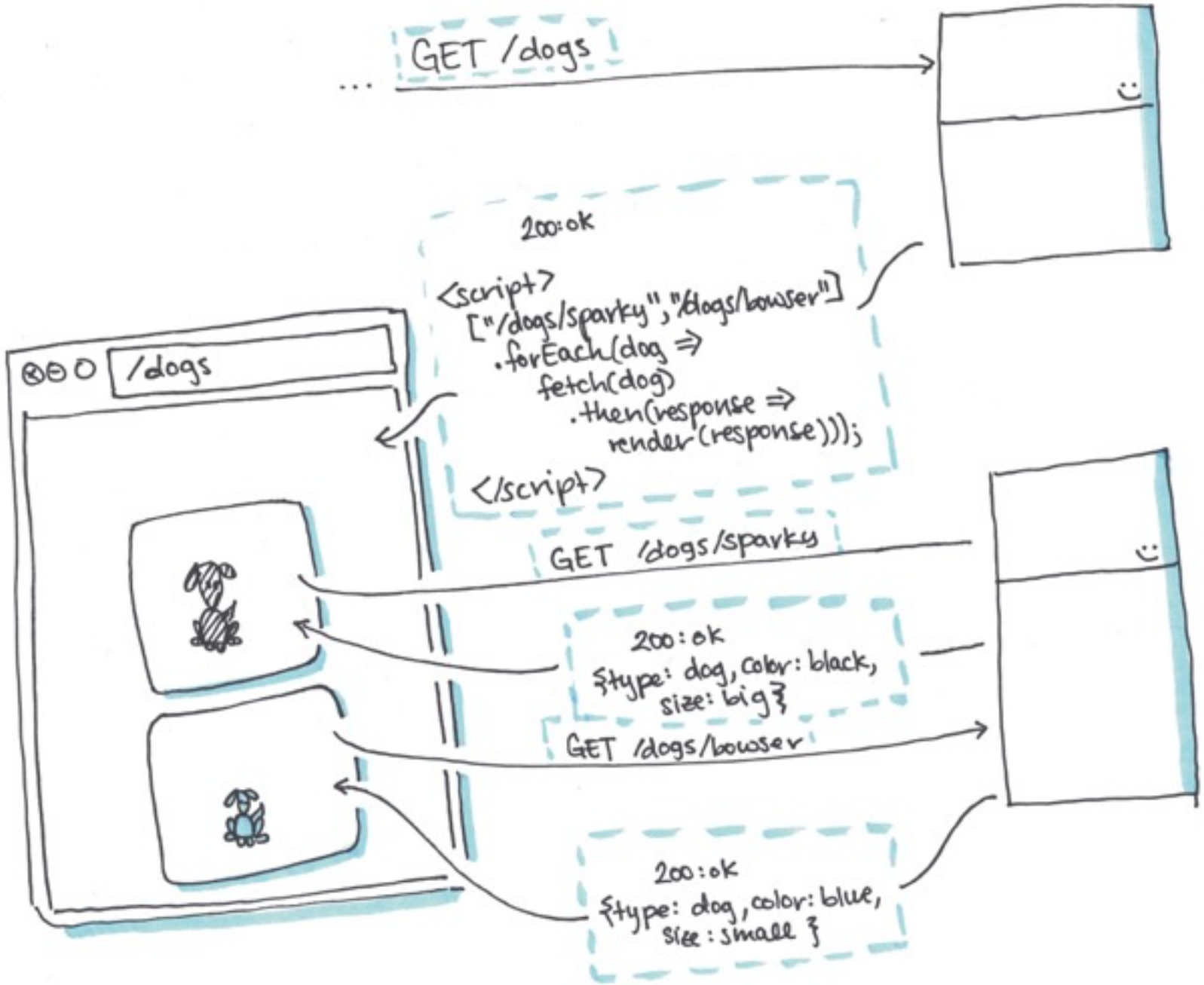
forms


GET /mysite
200: ok
<form
action="/go"
method="post">
<input type="text"
name="a">
<input type="text"
name="b">
<input type="submit">
</form>
/mysite
submit
/mysite
foo
bar
submit
POST /mysite?a=foo&b=bar

redirects


POST /dogs?big=true&color=black
302: found
/dogs/sparky
/dogs/sparky
GET /dogs/sparky
/dogs/sparky
200: ok

JavaScript


GET /dogs
200:ok
<script>
["/dogs/sparky","/dogs/bowser"]
.forEach(dog =>
fetch(dog)
.then(response =>
render(response)));
</script>
/dogs
GET /dogs/sparky
200:ok
{type: dog, color: black,
size: big}
GET /dogs/bowser
200:ok
{type: dog, color: blue,
size: small}

Clojure

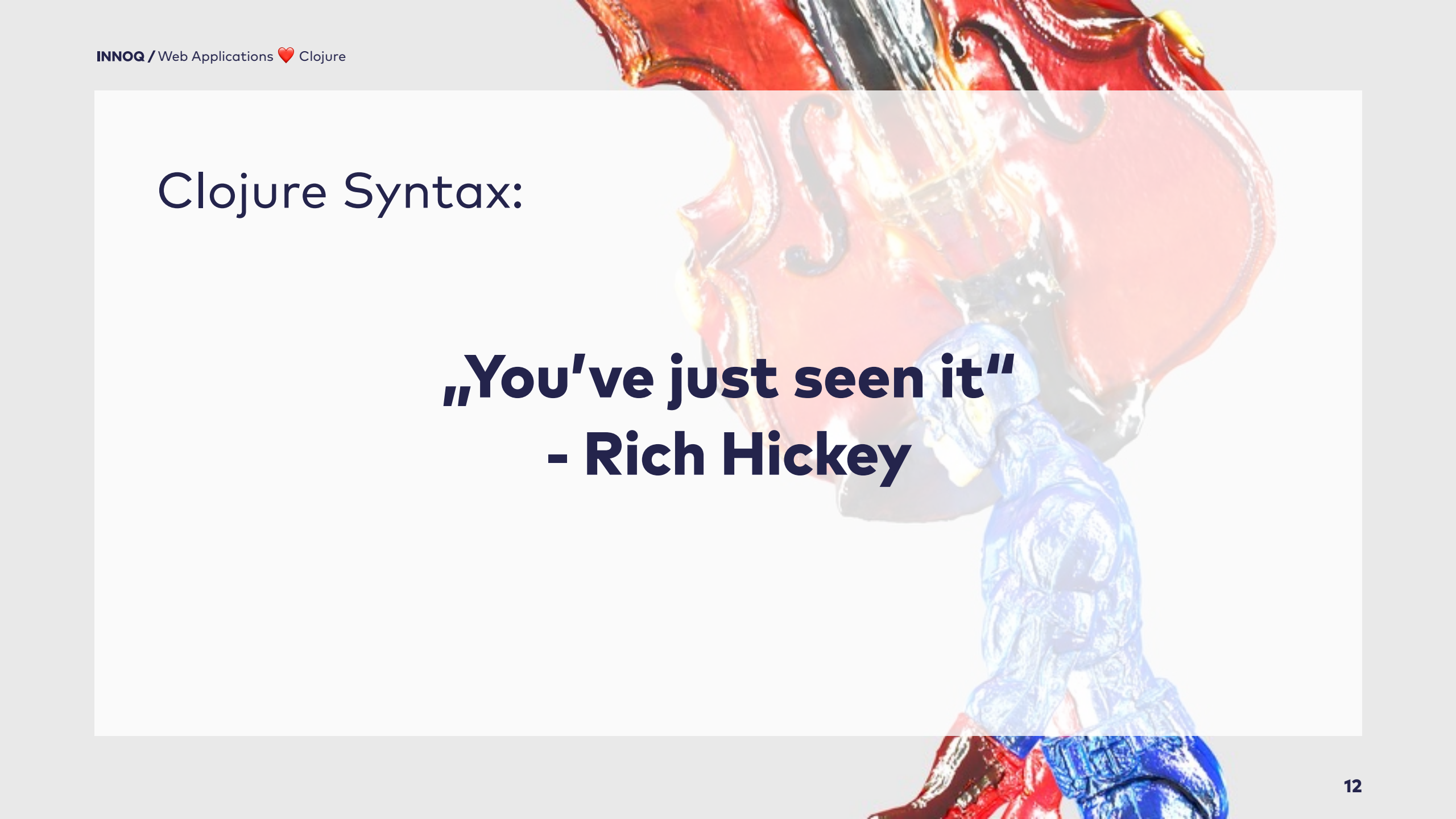
- Lisp Variant for the JVM
- Functional Language
- Dynamically Typed
- Immutable Data Structures
- Simplicity - Separation of Data and Behavior

Data Structures

```
3   3.14   3/2

                “Hello World”    \a    #“Ch.*se”

foo             :first                (“Hello” :first)

 {:name “Joy”                                [3 4 3]
  :company “INNOQ”}                        #{3 4 3}
```

Functions

```
(+ 1 2 3)
> 6


(:city {:name “INNOQ”
        :city “Berlin”})
> “Berlin”


(map inc [1 2 3])
> (2 3 4)
```



Clojure Syntax:

„You've just seen it"
- Rich Hickey

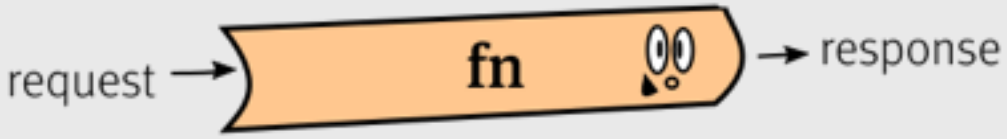
Web Applications + Clojure =

What's a Web Application?

Something that takes a

HTTP Request and returns a

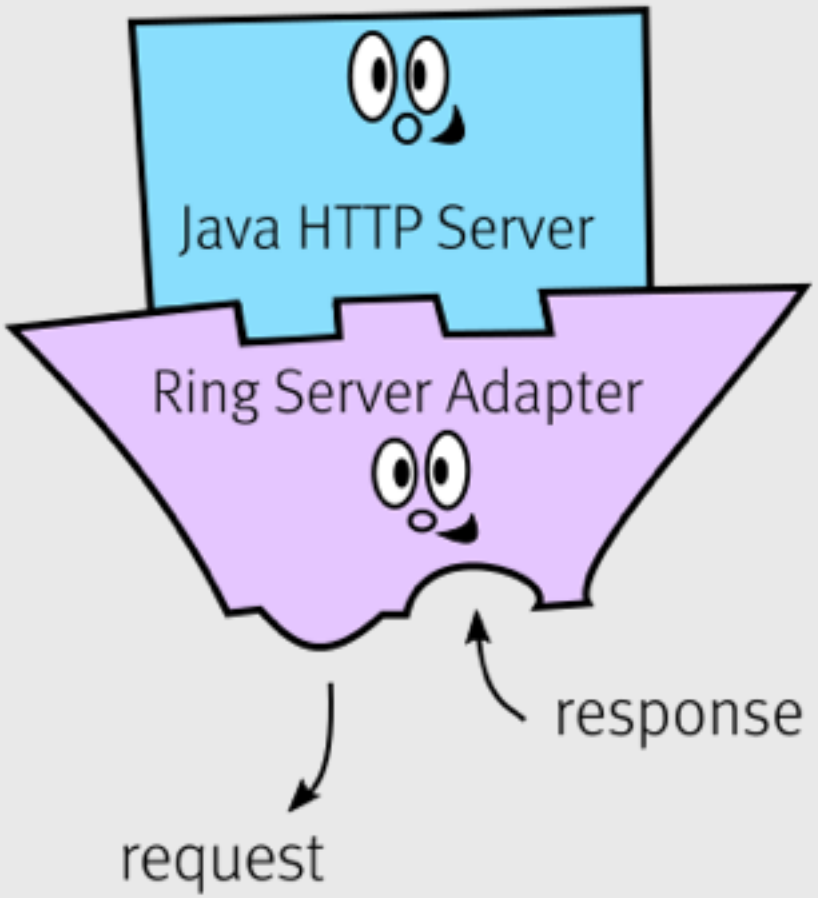
HTTP Response



Ring

https://github.com/ring-clojure/ring

- HTTP Server abstraction
- Request & Response are data
- Web App is a function



Ring: Request

```
{:uri “/”
  :request-method :get
  :headers { … }
  :params {:name “Joy”}
  …}
```

Ring: Response

```
{:status 200
 :headers { … }
 :body “Hello!”}
```

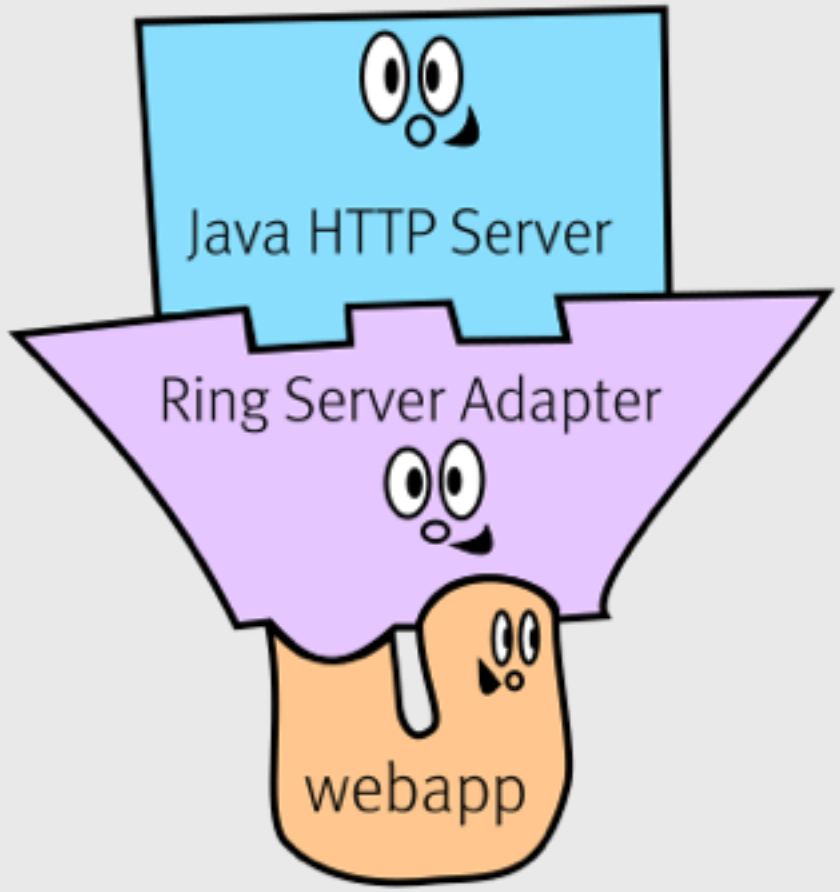
Ring: Handler

```
(defn example-app [request]
  (let [name (get-in request [:params :name])]
    {:status 200
     :headers {“Content-Type” “text/plain”}
     :body (str “Hello, ” name “!”)}))
```

Ring: Adapter

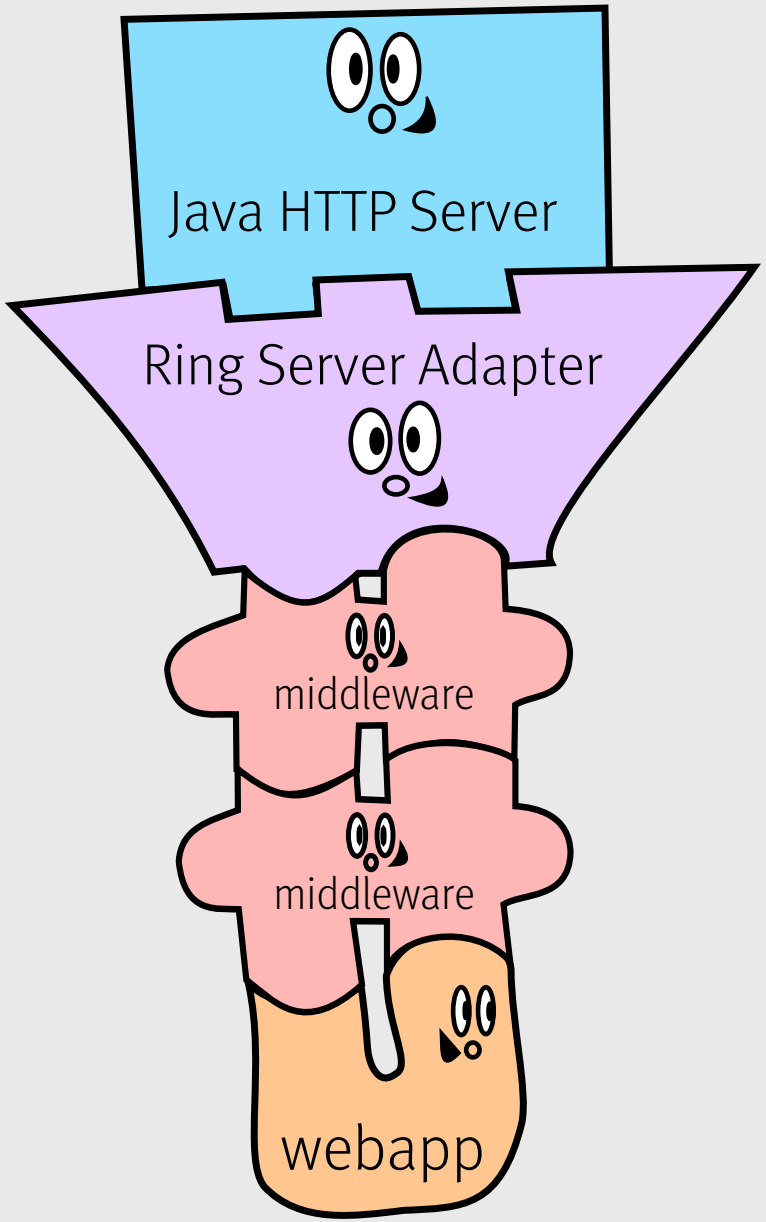
- Jetty
- Servlet
- http-kit
- Tomcat

...



Ring: Middleware

```
(defn wrap-logging [handler]
  (fn [request]
    (print request)
    (let [response (handler
                    request)]
      (print response)
      response)))
```



Ring: Middleware

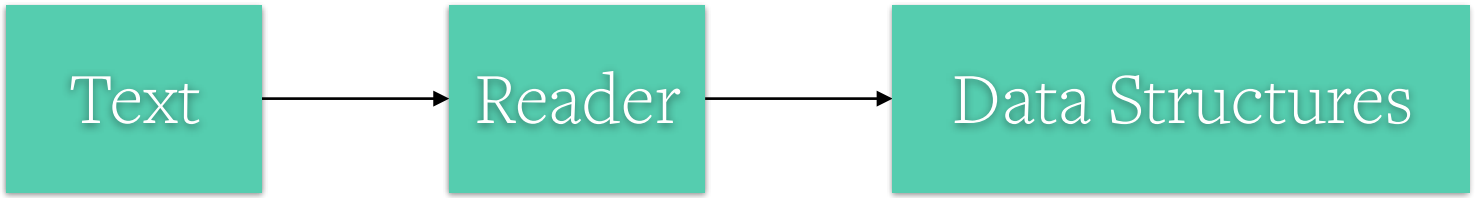
- https://github.com/ring-clojure/ring-defaults
- https://github.com/ring-clojure/ring-json

Compojure

https://github.com/weavejester/compojure

Excursion... Clojure Macros

Clojure: Macros




```
"(case 3
   1 "one"
   2 "two"
   "more")"
```

```
(case 3
  1 "one"
  2 "two"
  "more")
```

Clojure: Macros


```
(case
  1 “one”
  2 “two”
  “more”)
```

```
(if (= 3 1)
 “one”
 (if (= 3 2)
   “two”
   “more”))
```

Back to Compojure...

Compojure: Macro

```
(GET “/hello” [] “Hello, world!”)
```

```
(fn [req]
  (if (and (= (:uri req) “/hello”)
           (= (:request-method req) :get))
    {:body “Hello, world!”}))
```

Compojure: Handler

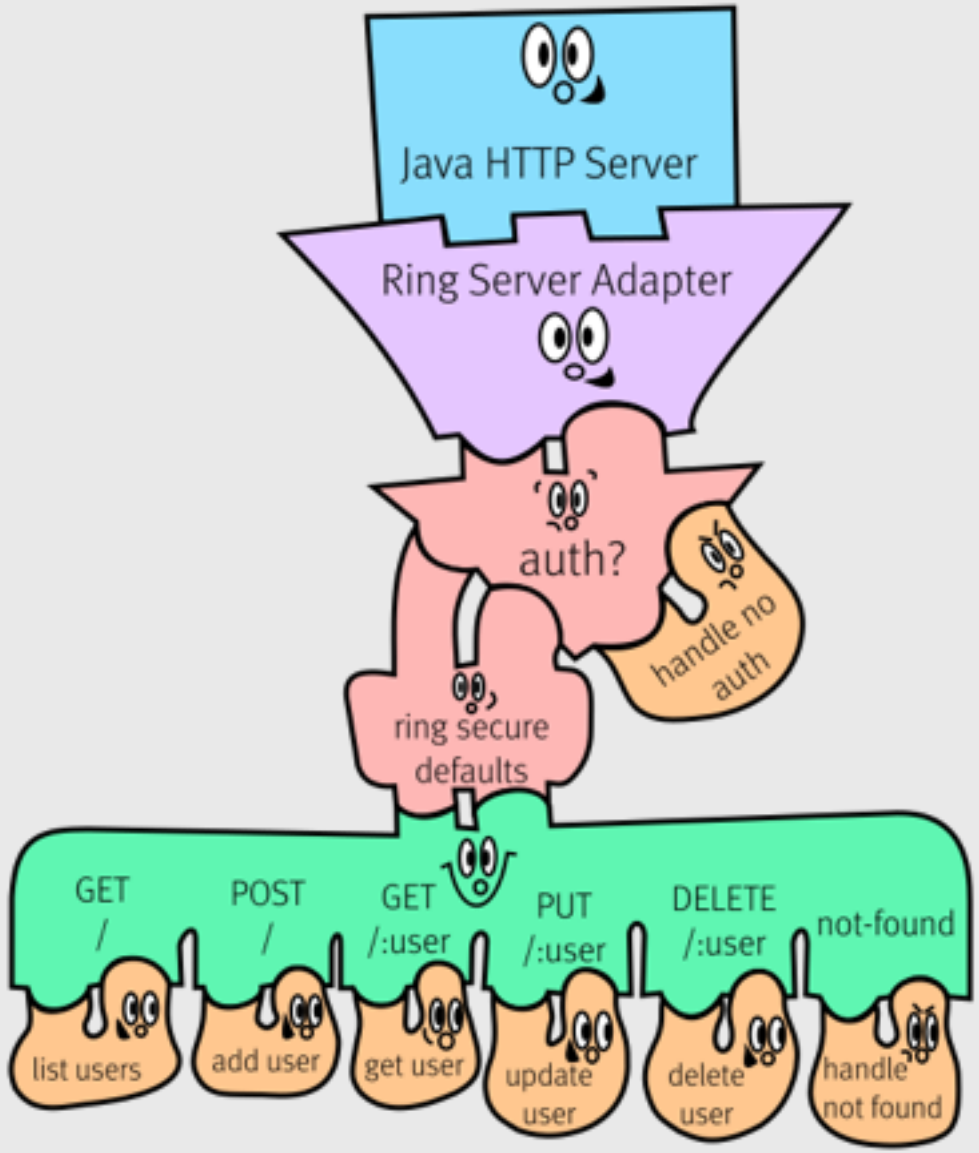
```
(defroutes app-routes
  (GET "/" request (list-users request))
  (POST "/" {params :params :as request}
    (add-user request params))
  (GET "/:username" [username :as request]
    (get-user request username)))
```

Compojure: Alternatives

- https://github.com/juxt/bidi
- http://clojure-liberator.github.io/liberator/

Putting it all together

Functional composition!



HTML Templating

<love> ❤️ </love>

Hiccup

```
<div id="foo">
  <span>bar</span>
</div>
```

```
[:div {:id "foo"}
  [:span "bar"]]
```

https://github.com/weavejester/hiccup

Hiccup

Make sure to escape HTML when using Hiccup!

```
(defn render-html [input]
  (hiccup.util/escape-html
    (hiccup.core/html input)))
```

Hiccup: Alternatives

- https://github.com/cgrand/enlive
- https://github.com/yogthos/Selmer

Summary

Summary

- Functional language ideal for Web Apps
- Clojure is a stable language
- Libraries vs. Frameworks
- Community is healthy and welcoming

Interesting Libraries

- https://github.com/technomancy/leiningen
- https://github.com/boot-clj/boot
- https://github.com/weavejester/environ
- https://github.com/krisajenkins/yesql
- https://github.com/stuartsierra/component

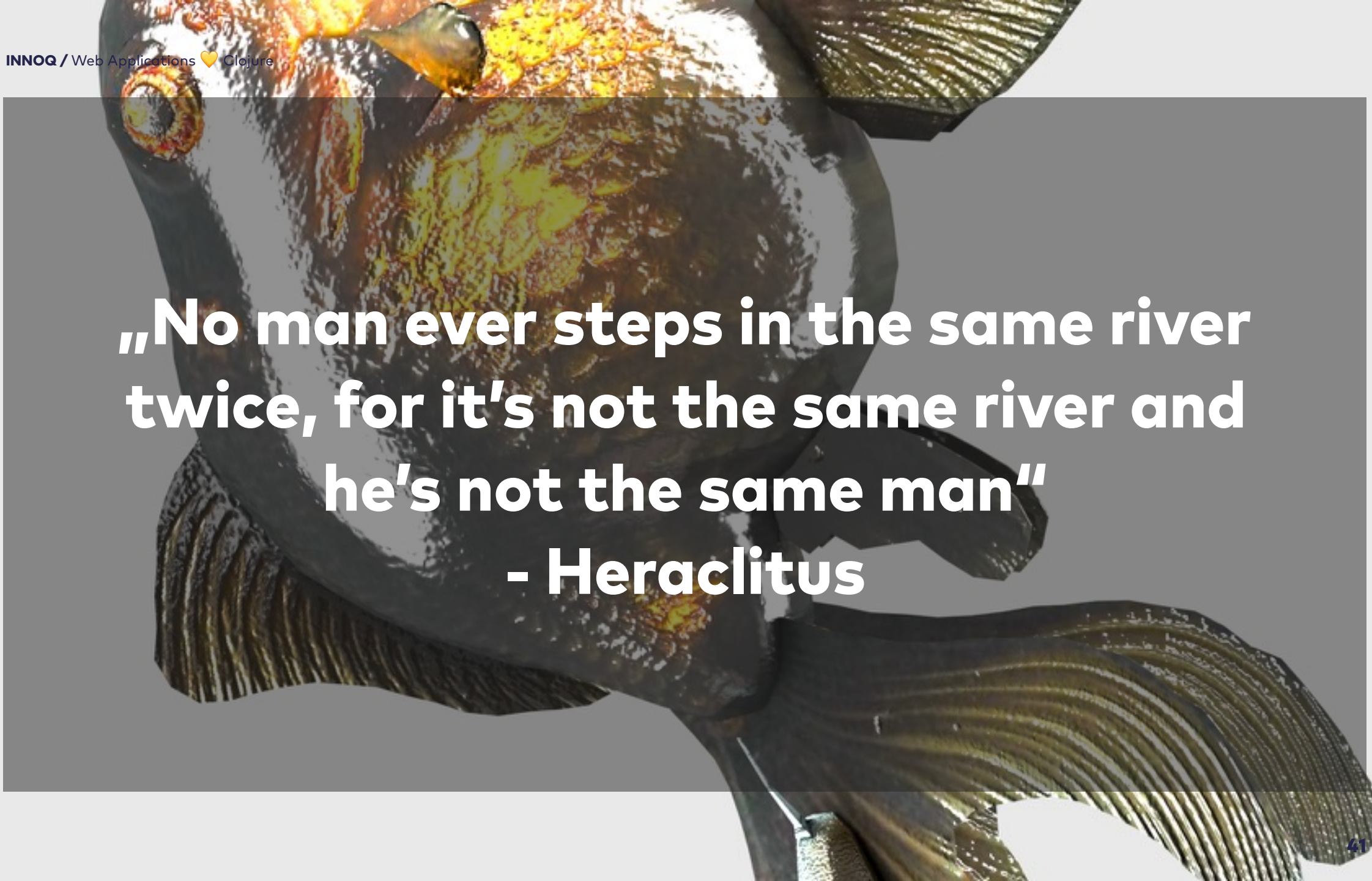
"Frameworks"

- https://github.com/weavejester/duct
- http://www.luminusweb.net
- https://github.com/metosin/compojure-api

Live Demo!

https://github.com/innoq/tutorial-clj-webapp

State in Clojure...

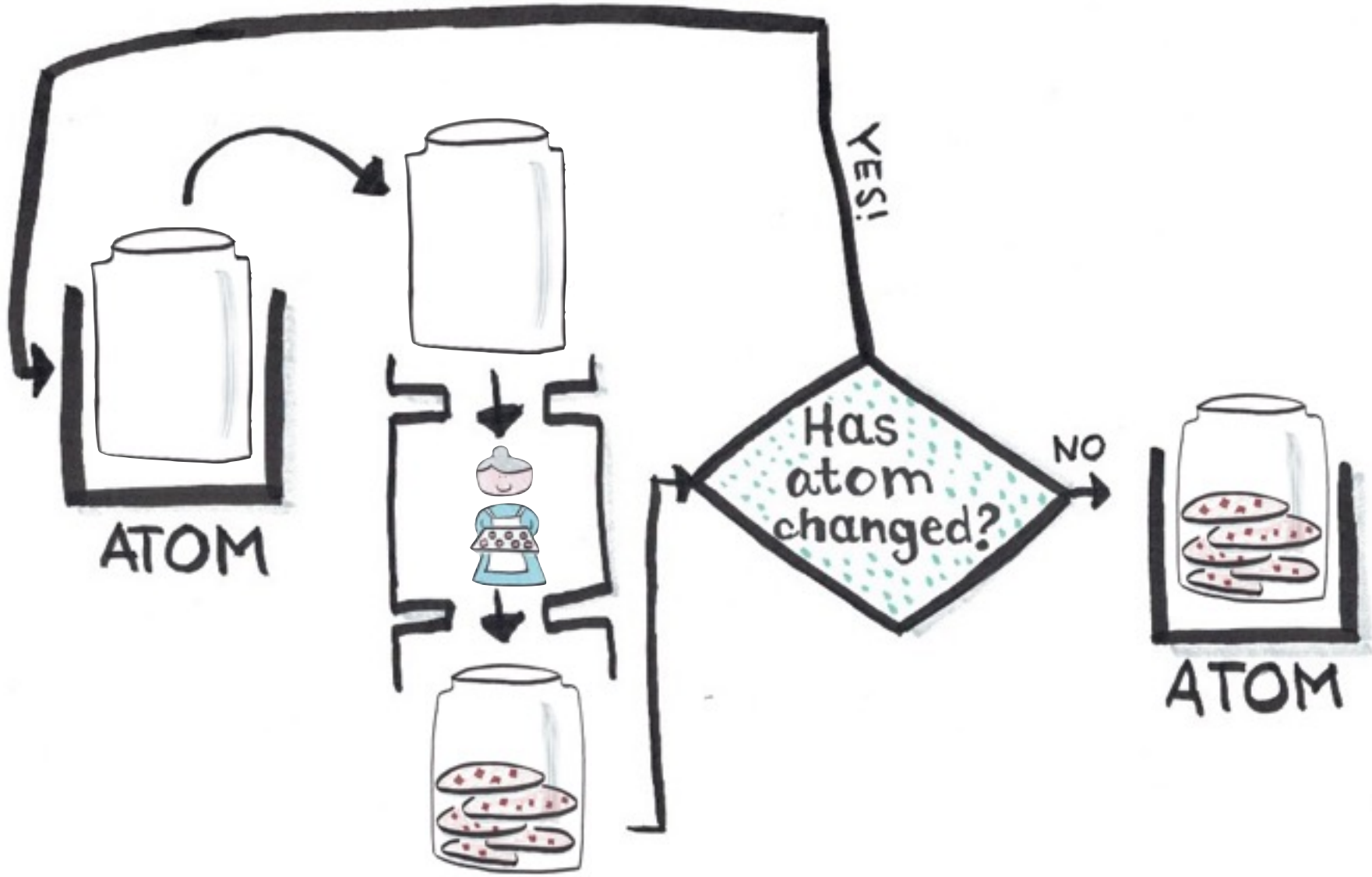


„No man ever steps in the same river twice, for it's not the same river and he's not the same man"
- Heraclitus

Clojure Identity

- The world around us is constantly changing.
- We shouldn't make the world "stop" so that we can change it.
- Instead...
  - We can give the world a function telling it what we want to change.

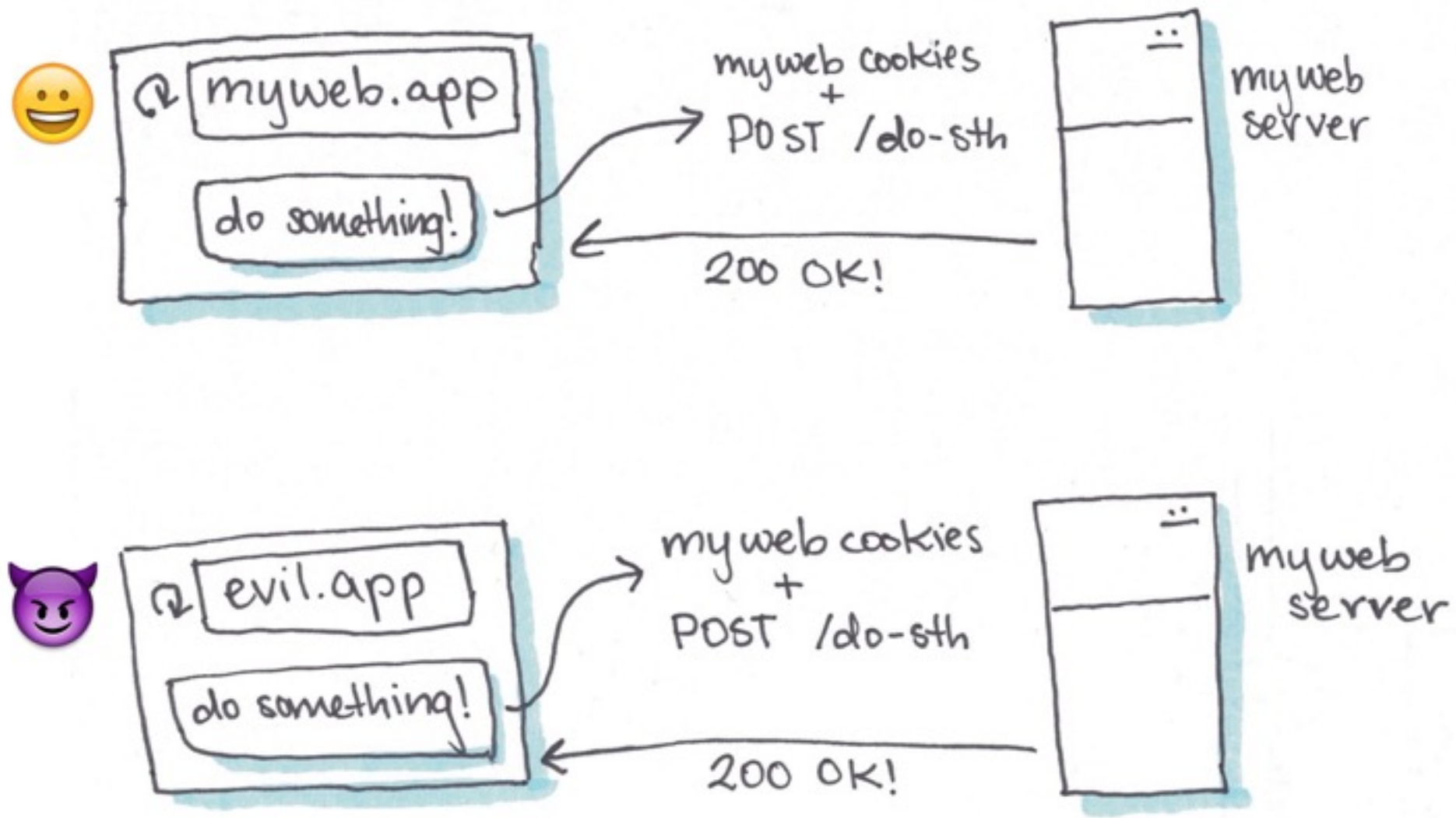
swap!



Joy Clark | @iamjoyclark

CSRF: Cross-Site Request Forgery

CSRF


myweb.app
do something!
myweb cookies
+
POST /do-sth
200 OK!
myweb server
evil.app
do something!
myweb cookies
+
POST /do-sth
200 OK!
myweb server

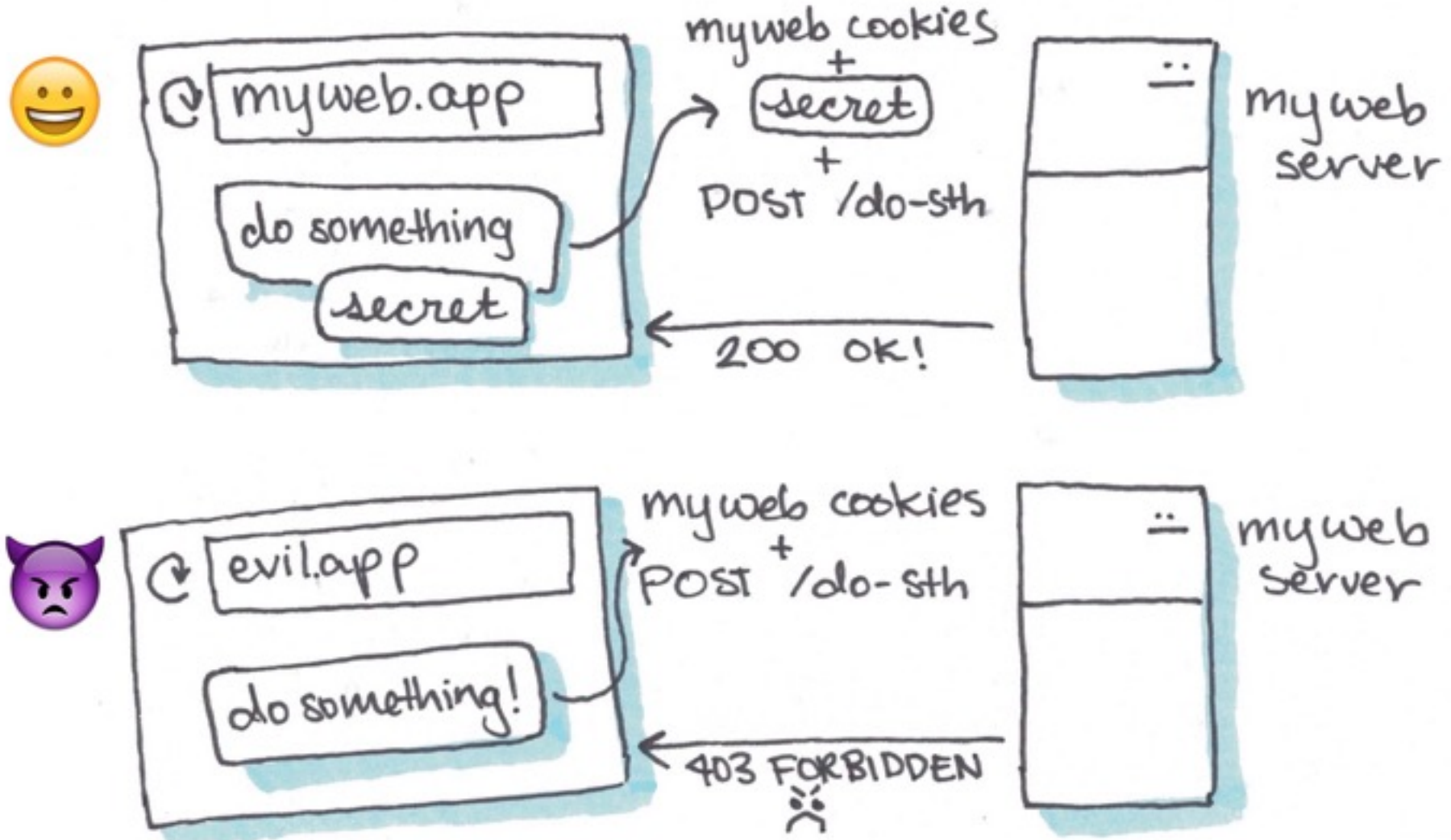
Protect Against CSRF

- Don't use GET requests to change state!

```
<img src="some http get request”>
```

- `SameSite=Strict` cookie attribute (default in site-defaults!)
- https://github.com/ring-clojure/ring-anti-forgery

CSRF Protection



Thank you!

Joy Clark | @iamjoyclark | joy.clark@innoq.com

https://github.com/innoq/tutorial-clj-webapp

INNOQ