Reagent Reference Sheet

https://purelyfunctional.tv/guide/reagent/

Form-1 Components

Just a function that returns Hiccup

```
(defn green-button [txt]
  [:button.green txt])
```

Will re-render when args change or when Reagent Atoms it `deref`s change

Form-2 Components

```
(defn counting-button [txt]
  (let [state (reagent/atom 0)]
    (fn [txt]
      [:button.green
        {:on-click #(swap! state inc)}
        (str txt " " @state)])))
```

argument lists should match or it won't re-render when they change

use Reagent Atom to store local state

Form-3 Components

```
(defn complex-component [a b c]
  (let [state (reagent/atom {})]
    (reagent/create-class
      {:component-did-mount
       (fn [this] (println "I mounted"))

       :display-name "complex-component"

       :render-reagent
       (fn [a b c]
         [:div {:class c}
           [:i a] " " b])})))
```

use `reagent/create-class` to create the component

define the lifecycle method you'd like

`this` argument is available to refer to the component

name will be included in error messages

note that you must define `:render-reagent` not `:render` to get the Reagent magic

Hiccup

```
[:div#login-form.row
  [:form {:method :post :action "/login"
          :style {:margin-left "10px"
                  :text-align :center}}
    [username-field]
    [password-field a b c]
    [:button {:type :submit
              :class ["btn" "btn-blue"]}
      "Log in"]]]
```

id and class shorthand `id="login-form" class="row"`

attributes go in a map

style attribute is broken out into a map instead of a string

embed components directly in `[]`

segs are allowed instead of string concatenation

Reagent Atoms

```
(defonce state (reagent/atom [1 2]))
(defn big-component []
  (let [[a b] @state]
    [:div
      [my-component a]]))
```

usually defined using `defonce` to allow for reloading

construct using `reagent/atom`

`deref` inside a component so they are re-rendered on changes to the atom

`my-component` is re-rendered when `a` changes

Seqs & Lazy seqs

```
(defonce student-urls (reagent/atom {}))
(defn student-list [students]
  [:ul
    (doall
      (for [student students]
        [:li {:key (:id student)}
          [:a {:href (get @student-urls (:id student))}
            (:name student)]]))])
```

`for` is lazy

components that render lists should add a `:key` attribute that is stable and unique

we `deref` inside the `for`

force evaluation to ensure component will re-render

PurelyFunctional.tv