Episode 2 Codes for printing

client.clj

```clojure
(ns undead.client
  (:require [quiescent.core :as q]
            [quiescent.dom :as d]))

(def game {:board [{:face :h1} {:face :h1} {:face :h2} {:face :h2 :revealed? true}
                   {:face :h3} {:face :h3} {:face :h4 :matched? true} {:face :h4 :matched? true}
                   {:face :h5} {:face :h5} {:face :fg} {:face :fg}
                   {:face :zo} {:face :zo :matched? true} {:face :zo :matched? true} {:face :gy}]
           :sand (concat (repeat 10 :gone)
                         (repeat 20 :remaining))
           :foggy? false})

(q/defcomponent Cell [cell]
  (d/div {:className "cell"}
         (d/div {:className (str "tile"
                                 (when (:revealed? cell) " revealed")
                                 (when (:matched? cell) " matched"))}
                (d/div {:className "front"})
                (d/div {:className (str "back " (name (:face cell)))}))))

(q/defcomponent Line [cells]
  (apply d/div {:className "line"}
         (map Cell cells)))

(q/defcomponent Board [cells]
  (apply d/div {:className "board clearfix"}
         (map Line (partition 4 cells))))

(q/defcomponent Timer [{:keys [sand index]}]
  (apply d/div {:className (str "timer timer-" index)}
         (map #(d/div {:className (str "sand " (name %))}) sand)))

(q/defcomponent Timers [sand]
  (apply d/div {}
         (map-indexed #(Timer {:index %1 :sand %2}) (partition 30 sand))))

(q/defcomponent Game [game]
  (d/div {:className (when (:foggy? game) "foggy")}
         (Board (:board game))
         (Timers (:sand game))))

(q/render (Game game)
          (.getElementById js/document "main"))
```