Episode 1 Codes for printing

project.clj

```
(defproject parens-of-the-dead "0.1.0-SNAPSHOT"
  :description "A series of zombie-themed games written with Clojure and ClojureScript."
  :url "http://www.parens-of-the-dead.com"
  :license {:name "GNU General Public License"
            :url "http://www.gnu.org/licenses/gpl.html"}
  :jvm-opts ["-XX:MaxPermSize=256m"]
  :main undead.system
  :dependencies [[org.clojure/clojure "1.7.0"]
                 [org.clojure/clojurescript "0.0-3308"]
                 [http-kit "2.1.18"]
                 [com.stuartsierra/component "0.2.3"]
                 [compojure "1.3.4"]]
  :profiles {:dev {:plugins [[lein-cljsbuild "1.0.6"]
                             [lein-figwheel "0.3.7"]]
                   :dependencies [[reloaded.repl "0.1.0"]]
                   :source-paths ["dev"]
                   :cljsbuild {:builds [{:source-paths ["src" "dev"]
                                         :figwheel true
                                         :compiler {:output-to "target/classes/public/app.js"
                                                    :output-dir "target/classes/public/out"
                                                    :optimizations :none
                                                    :recompile-dependents true
                                                    :source-map true}}]}}})
```

web.clj

```
(ns undead.web
  (:require [compojure.core :refer [defroutes GET]]
            [compojure.route :refer [resources]]))

(defn index [req]
  {:status  200
   :headers {"Content-Type" "text/html"}
   :body    "Hello from Compojure!"})

(defroutes app
  (GET "/" [] index)
  (resources "/"))
```

system.clj

```clojure
(ns undead.system
  (:require [com.stuartsierra.component :as component]
            [org.httpkit.server :refer [run-server]]
            [undead.web :refer [app]]))

(defn- start-server [handler port]
  (let [server (run-server handler {:port port})]
    (println (str "Started server on localhost:" port))
    server))

(defn- stop-server [server]
  (when server
    (server))) ;; run-server returns a fn that stops itself

(defrecord ParensOfTheDead []
  component/Lifecycle
  (start [this]
    (assoc this :server (start-server #'app 9009)))
  (stop [this]
    (stop-server (:server this))
    (dissoc this :server)))

(defn create-system []
  (ParensOfTheDead.))

(defn -main [& args]
  (.start (create-system)))
```

client.clj

```clojure
(ns undead.client)

(js/alert "OOOOH YEAH")
```