CS 696 Functional Programming and Design
Fall Semester, 2015
Doc 20 Starting a Reagent Project
Nov 12, 2015

Copyright ©, All rights reserved. 2015 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/openpub/) license defines the copyright on this document.
Blog & Video

Curious about ClojureScript, but not sure how to use it


You task for Tuesday:

Implement Tick-tack-toe from the video
Creating Reagent Project

Single View

`lein new figwheel projectName -- --reagent`

Single Page App with multiple views

`lein new reagent projectname`
lein & Creating Projects

lein new $TEMPLATE_NAME $PROJECT_NAME

lein new reagent projectname

Arguments can be passed to templates by adding them after "new"'s options. Use `--` to separate arguments to lein new and the actual template you are using:

lein new figwheel projectName -- --reagent
Figwheel

Builds your ClojureScript code and hot loads it into the browser as you are coding!

To start Figwheel in a terminal inside your project directory

lein figwheel
Single Page App

lein new figwheel tetris -- --reagent
(ns ^:figwheel-always tetris.core
   (:require 
     [reagent.core :as reagent :refer [atom]]))

(enable-console-print!)
(println "Edits to this text should show up in your developer console.")

(defonce app-state (atom {:text "Hello world!"]))

(defn hello-world []
  [:h1 (:text @app-state)])

(reagent/render-component [hello-world]
  (. js/document (getElementById "app")))

(defn on-js-reload []
  ;; (swap! app-state update-in [:__figwheel_counter] inc)
  )
Starting a Development Server

In terminal type "lein figwheel"

AI pro 5->lein figwheel
Figwheel: Starting server at http://localhost:3449
Browser at 3449

Hello world!
Demo

Show changes
Single Page App with multiple views
First Project

lein new reagent projectname
(def mount-target 
[:div#app 
 [:h3 "ClojureScript has not been compiled!"
 [:p "please run 
 [:b "lein figwheel"] " in order to start the compiler"]])

(def loading-page 
(html 
 [:html 
 [:head 
 [:meta {:charset "utf-8"}] 
 [:meta {:name "viewport" 
 :content "width=device-width, initial-scale=1"}] 
 (include-css (if (env :dev) "css/site.css" "css/site.min.css")))] 
 [:body 
 mount-target 
 (include-js "js/app.js")]))

(defroutes routes 
 (GET "/" [] loading-page) 
 (GET "/about" [] loading-page)
(defroutes routes
  (GET "/" [] loading-page)
  (GET "/about" [] loading-page))

- Method: GET, POST
- URL
- URL parameters
- Function to call
- When URL is requested
(def loading-page
  (html
   [:html
    [:head
     [:meta {:charset "utf-8"}]
     [:meta {:name "viewport"
            :content "width=device-width, initial-scale=1"]
     (include-css (if (env :dev) "css/site.css" "css/site.min.css"))]
    [:body
     mount-target
     (include-js "js/app.js")])))
(def mount-target
  [:div#app
    [:h3 "ClojureScript has not been compiled!"
    [:p "please run 
      [:b "lein figwheel"
      " in order to start the compiler"]])}
Client-Side Libraries

accountant.core

ClojureScript library to make navigation in single-page applications simple

secretary.core

Defines client side routes
  URLs & function to call

reagent.session

Just an atom
  Used to store state
(defn home-page []
  [:div [:h2 "Welcome to foobar"]
   [:div [:a {:href "/about"} "go to about page"]]])

(defn about-page []
  [:div [:h2 "About foobar"]
   [:div [:a {:href "/"} "go to the home page"]]])

(defn current-page [] [:div [(session/get :current-page)]])

(secretary/defroute "/" []
  (session/put! :current-page #"home-page")

(secretary/defroute "/about" []
  (session/put! :current-page #"about-page")

(defn mount-root []
  (reagent/render [current-page] (.getElementById js/document "app")))

(defn init! []
  (accountant/configure-navigation!)
  (accountant/dispatch-current!)
  (mount-root))
Hiccup for HTML

(defn home-page []
  [:div [:h2 "Welcome to foobar"]
   [:div [:a {:href "/about"} "go to about page"]]]
)

(defn about-page []
  [:div [:h2 "About foobar"]
   [:div [:a {:href "/"} "go to the home page"]]])
Routes

(secretary/defroute "/" []
  (session/put! :current-page #'home-page))

(secretary/defroute "/about" []
  (session/put! :current-page #'about-page))

For each URL
Change atom to hold reference to which function to call
(defn current-page [] [:div [(session/get :current-page)]])

Lists are expanded in Hiccup
So expands to the current page

(defn mount-root []
  (reagent/render [current-page] (.getElementById js/document "app")))

Magic function
Render the client page each time current-page changes