Web Development
<!DOCTYPE html>
<html lang="en">
<head>
<meta>
<title>
Sample
</title>
</head>
<body>
<h1>
Hello World
</h1>
<p>
This is a small page
</p>
</body>
</html>
Issues

Presentation

Dynamic generation

Client side interaction
Presentation - CSS

Cascading Style Sheets

Use CSS for layout
Use HTML for structure

http://www.w3schools.com/css/
http://www.csszengarden.com/219/
http://www.csszengarden.com/217/

The Beauty of CSS Design

Select a Design:
- Mid Century Modern by Andrew Lohman
- Garments by Dan Mall
- Steel by Steffen Knoeller
- Apothecary by Trent Walton
- Screen Filler by Elliot Jay Stocks
- Fountain Kiss by Jeremy Carlson
- A Robot Named Jimmy by meltmedia
- Verde Moderna by Dave Shea

Archives:
Dynamic Generation of HTML

List of animals in database

How to create an HTML list

<ol>
  <li>Bat</li>
  <li>Cat</li>
  <li>Dog</li>
  <li>Elephant</li>
</ol>
## Embed a language in HTML

Template System

PHP
Selmer

{% for ex in exams %}
<tr>
  <td>{{ex.title}}</td>
  <td>{{ex.exam_date}}</td>
  <td>{{ex.exam_start_time}} - {{ex.exam_end_time}}</td>
  <td>{{ex.exam_location}}</td>
  <td>{{ex.term}}</td>
  <td>{{ex.register_start}} to {{ex.register_end}}</td>
  <td>{{ex.pass_grade}}</td>
</tr>
{% endfor %}
Generate HTML in your Programming Language

(h/html [:ol
    (for [x animals]
        [:li x]])]

<ol>
    <li>Bat</li>
    <li>Cat</li>
    <li>Dog</li>
    <li>Elephant</li>
</ol>
HTML
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta>
    <title>
      Sample
    </title>
  </head>
  <body>
    <h1>
      Hello World
    </h1>
    <p>
      This is a small page
    </p>
  </body>
</html>
Tags & Attributes

<a href="http://www.eli.sdsu.edu">Me</a>

<img src="w3schools.jpg" width="104" height="142">

<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>

<p>This is<br>a para<br>graph with line breaks</p>

<body style="background-color:lightgrey">

<p><b>This text is bold</b>.</p>

<i>Wednesday, November 10, 15</i>
HTML Tutorial

http://www.w3schools.com/html/default.asp
CSS

<!DOCTYPE html>
<html>
<head>
<style>
body {background-color:lightgrey}
h1   {color:blue}
p    {color:green}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
div - Container element

<div style="background-color:black; color:white; padding:20px;">
    <h2>Hello</h2>
    <p>Here is a sentence</p>
</div>
CSS classes

HTML elements can have multiple classes

```html
<!DOCTYPE html>
<html lang="en">
  <head>
  </head>
  <body>
    <p>A Clojure example</p>
    <p class="code">(-> 1000 factorial str count)</p>
  </body>
</html>
```
Attribute id

ids should be unique in a document

<p id="sam">2</p>
<a id="pete">4</a>

p#sam { color: lightblue }
CSS Tutorial

http://www.w3schools.com/css/
Hiccup
Hiccup

Clojure library representing HTML in Clojure

Vectors represent elements

Maps represent an element's attributes

[:p "Hello World"]

[:a {:href "http://www.eli.sdsu.edu"} "Me"]
Basic Syntax

[tag & body]
[tag attributes & body]

First item must be a tag
Second item can be map
  Attribute

Body
  String
  Nested tag vectors

[:a {:href "http://www.eli.sdsu.edu"} "Me"]

[:p "Hello " [:em "World!"]]
<table>
<thead>
<tr>
<th>Tags</th>
<th>Symbol</th>
<th>String</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>['p &quot;Hello &quot; &quot;World&quot;]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>['&quot;p&quot; &quot;Hello &quot; &quot;World&quot;]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[:p &quot;Hello &quot; &quot;World&quot;]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Expanding seqs

seqs are expanded in body

[:div (list "Hello" "World")]
[:div "Hello" "World"]

[:p (if (< x 5) "Hi" "bye")]
[:p "Hi"] or [:p "bye"]

[:p [(if (< x 5) :em :italic) "cat"]]
[:p [:em "cat"]]
[:p [:italic "cat"]]

(def animals ["Bat" "Cat" "Dog" "Elephant"])
(h/html [:ol
  (for [x animals]
    [:li x]])
Class & ID

[:div {:class "selected"} [:p "hi"]]

[:div.selected [:p "hi"]]

[:div {:id "pete"} [:p "hi"]]

[:div#pete [:p "hi"]]

Tuesday, November 10, 15
Single Page Application
SPA
Document Object Model (DOM)

Web browsers convert HTML page into a tree of objects

Objects have methods & can be changed

DOM is used to display the page
DOM & JavaScript

JavaScript can add, change, and remove all the HTML elements and attributes in the page
JavaScript can change all the CSS styles in the page
JavaScript can react to all existing events in the page
JavaScript can create new events in the page

Single Page Application (SPA)

Web application that fits on a single web page

All required resources (HTML, CSS, JavaScript) loaded
  First page load
  In background as needed

More native app like experience
  No round trip requests for next page
  Menus, buttons, drag&drop handled on client
  Client can redraw any part of the screen

Supports multiple URLs
  Bookmarking
  Google indexing
Reagent
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