DESIGN REPORT #4

SPATIAL & ENCYCLOPEDIC AFFORDANCES • THE POKÉSPACE

#APIS, #DYNAMIC_PAGES, #SPATIAL & #ENCYCLOPEDIC AFFORDANCES, #CSS_FRAMEWORKS, #MODEL-HUMAN PROCESSOR

DEADLINES

Module Release	March 25
Module Due Date	April 10, End of Day
Module Exam	April 10-12, End of Day

[Inara Rupani]

[https://us-east-2.console.aws.amazon.com/cloud9/ide/7c94b0cd46cc41dabe433772de24dd4a
[https://docs.google.com/document/d/1gXeaWgBe5jp9zirW6nkG-5x3k_87-bpAhvYFiT6svqU/edit?usp=sharing]
[https://www.figma.com/file/2ub35Ea5JTzdOijU4YYyo7/Module-4-IR?node-id=1%3A5]

TEACHING OBJECTIVES

- Working knowledge of CSS frameworks
- Collect, process, and inject data into HTML pages.
- Utilize **asynchronous calls to third-party APIs** as data sources.
- Apply and appraise **spatial and encyclopedic affordances** in UI design.

DESIGN BRIEF (100 PTS TOTAL)

One of the most powerful affordances of the computer is the ability to create worlds. In 2016, when Pokemon Go was released, the world experienced this affordance play out in the physical world, traversing the physical and virtual worlds and unlocking a rich new space of augmented reality interactions. While geotagging and geofencing were common AR and GIS interactions, Pokemon Go also accessed a rich information space of the Pokemon. For this assignment, we will leverage the Pokemon dataset to enhance AR game interactions.

This assignment consists of two technical vignettes (piecewise technical implementation problem sets) aimed to develop your web programming skill set.

> The first uses a CSS framework known as semantic to markup HTML and generate attractive and usable user interfaces. We will be developing a HTML/CSS mockup of a “Pokedex” app.
>
> The second is about learning to use three libraries to create dynamic and interactive content: **jQuery, underscore,** and **AJAX** (we will be using jQuery’s $.ajax helper function). We will use these to call on **Google APIs** to acquire some useful services for maximizing spatial affordances.

The final and **most important part** of this module is proposing an interaction that maximizes encyclopedic and spatial affordances.

RESOURCES

- HTML/CSS: https://www.youtube.com/watch?v=UB1O30fR-EE
- JavaScript: https://www.codecademy.com/learn/introduction-to-javascript
- jQuery: https://www.udacity.com/course/intro-to-jquery--ud245 (highly recommended before beginning this module)

0. SETUP

1. Open your Cloud9 instance from Module 3.
2. In cloud9 Terminal, run:
    a. git clone https://github.com/CSE3392-S2020/module4-pokespace
    b. cd module4-pokespace
    c. bundle install
3. Update your run configuration CWD to point to the pokespace directory.
    a. https://www.youtube.com/watch?v=ZD7nHmk0mv8
    b.

SEMANTIC UI (20 PTS)

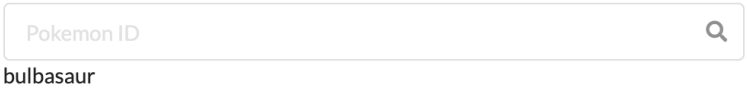
View	**vignettes/pokedex**
API	**https://pokeapi.co/pokemon/1**

Replicate the following UI using the **semantic-ui** CSS framework. A skeleton has been provided for you. Please note that all required libraries have been pre-imported. The right column has some information on where to find the specific semantic UI component.

You should only be changing the HAML code and very little CSS.

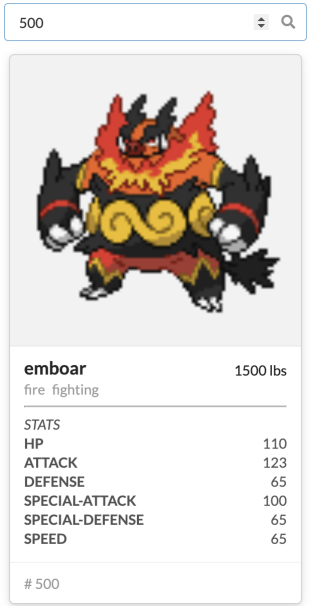
You will need to use the semantic framework classes:
Card
List

Your starting point:

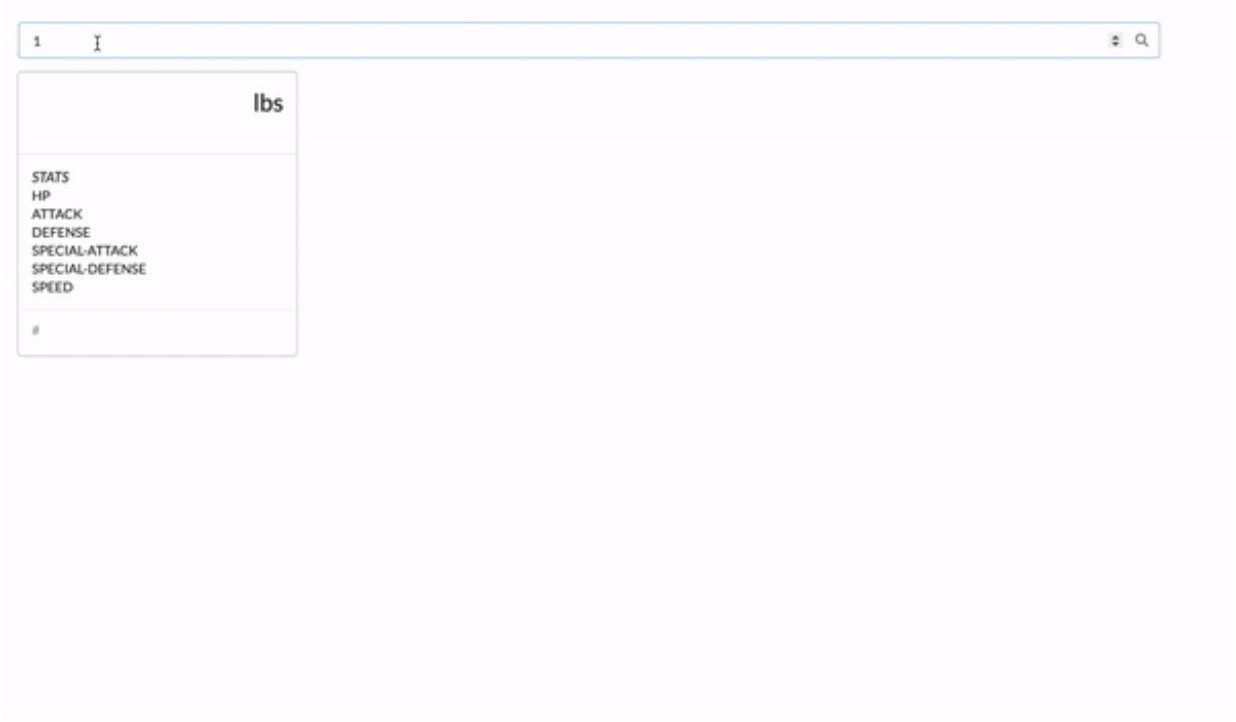


Example Goal 1: By changing the input to 62, the following should load on enter.

Example Goal 2: By changing the input to 500, the following should load on enter.



[TODO: GIPHY submittingPokemon IDs 14, 145, 168]



Link: https://gph.is/g/Z8d3M0b

[PASTE CLOUD9 CODE FOR POKEDEX (FORMAT USING CODEBLOCK AS HAML)]

```
.ui.card
        .image
          %img.ability.load{"data-query":"sprites.front_default","data-input":"src"}
        .content
          %div
            %span
              %h1{:style => "float: right;font-weight:normal; font-size:25px"}   lbs
            %span.right.floated
              %h1.weight.load{:style => "float: right;font-weight:normal;font-size:25px","data-query":"weight",
"data-input":"html"}
            %span
              %h1.name.load{:style => "float: left","data-query":"name", "data-input":"html"}

        %span.meta
          %p.type1.sameLine.load{:style => "display: inline-block; padding-left: 15px; font-size:
20px","data-query":"types.0.type.name","data-input":"html"}
          %p.type2.sameLine.load{:style => "display: inline-block; font-size:
20px","data-query":"types.1.type.name","data-input":"html"}

        .content
          %div
            %i{:style => "font-size: 15px; font-weight: lighter"} STATS
          .ui.list
            .item
              %p.stat.hp.load{:style => "font-size: 15px","data-query":"stats.0.base_stat", "data-input":"html"}
              %p.stat.label{:style => "font-size: 15px"} hp
            .item
              %p.stat.attack.load{:style => "font-size: 15px","data-query":"stats.1.base_stat", "data-input":"html"}
              %p.stat.label{:style => "font-size: 15px"} attack
            .item
              %p.stat.speattack.load{:style => "font-size: 15px","data-query":"stats.3.base_stat",
```

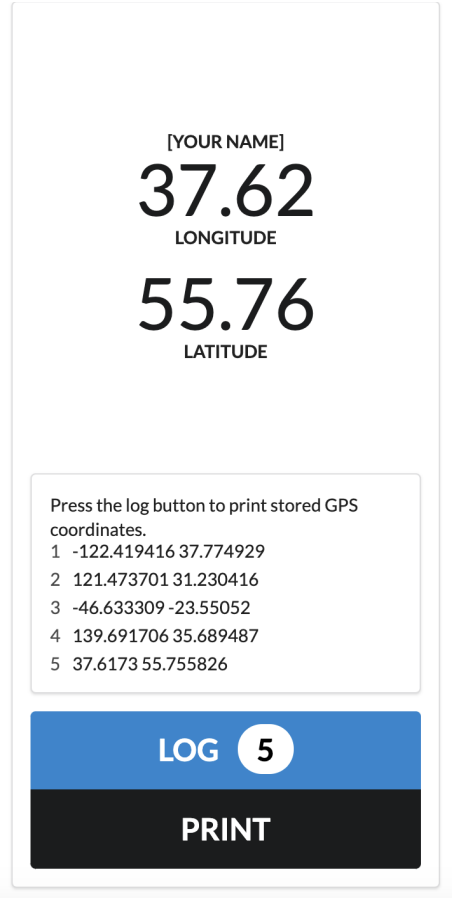
```
"data-input":"html"}
              %p.stat.label{:style => "font-size: 15px"} special-attack
            .item
              %p.stat.spedefense.load{:style => "font-size: 15px","data-query":"stats.4.base_stat",
"data-input":"html"}
              %p.stat.label{:style => "font-size: 15px"} special-defense
            .item
              %p.stat.speed.load{:style => "font-size: 15px","data-query":"stats.5.base_stat", "data-input":"html"}
              %p.stat.label{:style => "font-size: 15px"} speed

        .extra.content
          %p{:style => "display: inline-block"} #
          %p.number.load{:style => "display: inline-block","data-query":"id","data-input":"html"}
```

Check ++ Criteria:

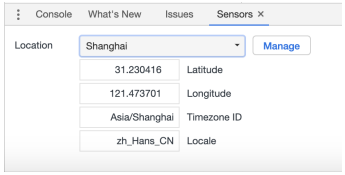
- Color code the different Pokemon types (e.g., green card for grass types, etc...)
- Load a blank card with an error message for invalid Pokemon IDs (e.g., -1)
- Redesign the Pokemon card to include more information.
- Allow the user to click through the different sprites of the Pokemon

GIS INTERACTIONS (50 PTS)

You will only be writing in CoffeeScript for this problem. ***Do not change the HTML or CSS code.***
Use the following **Google API Key**: AIzaSyAArgcv3N8XbwmrjebEtAWqbSdoNeUs-sg
API Key will become inactive after the assignment due date + 3 days..

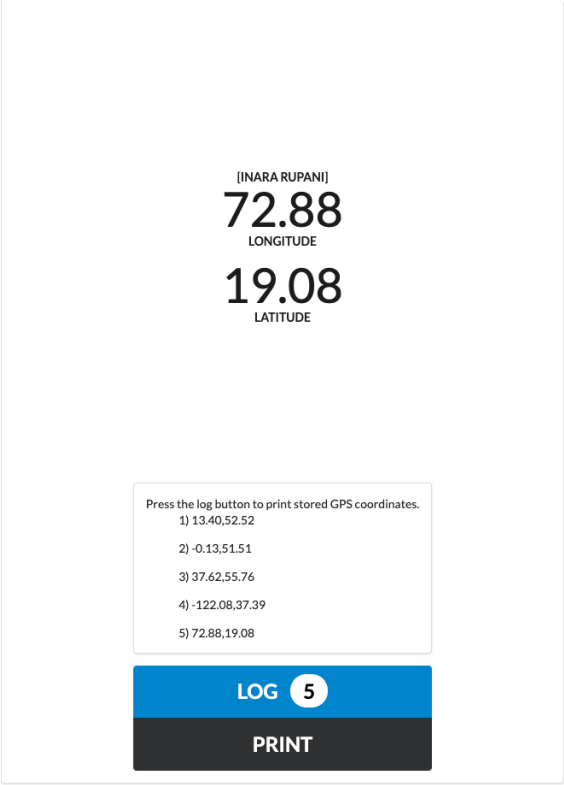
GEOCODING + PROGRAMMATIC DOM (25 PTS)	
View	**vignettes/geocode**
Example Output (values will be different)  	**a.** Do not alter the HAML/CSS for this assignment. i. Exception -- Update [YOUR NAME] with your name. **b.** Display continuous values of longitude and latitude on your device using the HTML5 GeoLocation API. i. Use a 3000 ms timeout value. ii. Enable high accuracy GPS coordinates. iii. Log error messages to the console using **console.error** or **console.warn** iv. Pretty-format longitude and latitude values to have 2 significant digits 1. Use the **toFixed** method. 2. Use jquery **.html(val)** to change the contents of DOM elements. **c.** When a user clicks the **LOG** button, the most recent coordinates are pushed into a global ***window.log*** array. i. Do not log invalid or undefined coordinates. ii. The badge element on the button UI should reflect the length of the current log. 1. Use jquery .html(val) to change the contents of DOM elements. d. When a user clicks the **PRINT** button, the contents of the log array are added as items in the **#output** list. i. Use the **underscore _.each** function to iterate through the log array. ii. Use the $("<tag>") to programmatically create HTML elements.

iii. Use the jquery **.append()** method to inject content into the DOM tree.
iv. Clear the previous contents of the list using jquery method **.html(“”).**

e. Using GIPHY, record yourself doing the following:
   i. Use the Console Tools > ... > Sensors tab to manually set GPS locations
      1. 

   ii. Log the following GPS locations
      1. Berlin
      2. London
      3. Moscow
      4. Mountain View
      5. Mumbai
   iii. Print the GPS locations

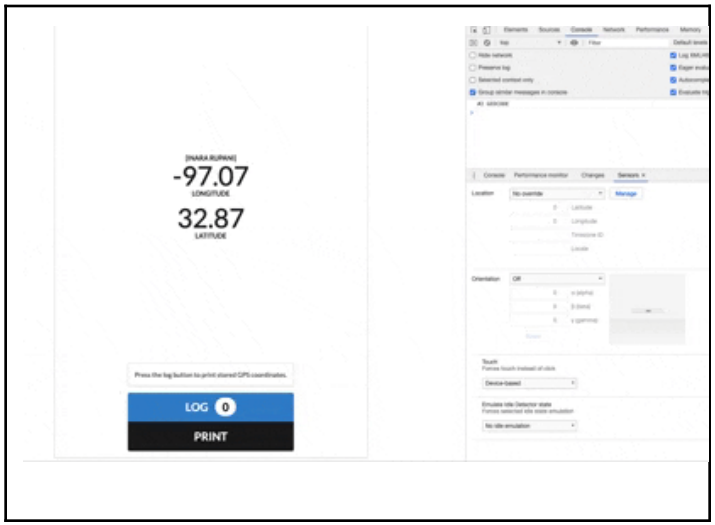
[Insert screenshot of result]



[Insert GIPHY here]

Link: https://gph.is/g/4LmwXYP

```
:coffeescript
  window.log = []
  long = undefined
  lat = undefined

  $ ->

    console.log("#2 GEOCODE")

    $('#log').click ->
      # console.log("log button clicked")
      x = [long, lat]
      window.log.push(x)
      console.log("logged x: " + x)
      $('.badge').html(window.log.length)

    $('#print').click ->
      # console.log("print button clicked")
      $('#output').html('')
      $.each window.log, (index, values)->
        console.log("printed: " + values)
        coords = $("<ol>").addClass("ui list ordered").html((index+1) +
') ' + values)
        $('#output').append(coords)

    initGeolocation = ->
      # console.log(navigator)
      if navigator and navigator.geolocation
        navigator.geolocation.watchPosition(successCallback,
errorCallback)
      else
        console.log 'Geolocation is not supported'
      return
```




```
successCallback = (position) ->
  long = position.coords.longitude.toFixed(2)
  lat = position.coords.latitude.toFixed(2)

  # DOM injection
  lo = $(".value:nth-child(2)").html(long)
  la = $(".value:nth-child(1)").html(lat)

  return

errorCallback = ->
  console.error ("ERROR")


watchID = navigator.geolocation.watchPosition(successCallback,
errorCallback, enableHighAccuracy: true, timeout: 3000)
```

POKEMON SIGHTINGS (25 PTS)	
View	**vignettes/sightings**
Example Output (values will be different)	2. **IMPORTANT: Update line 3 with the following API key** a. AIzaSyA5jvS0kjANVuGMZsswMqabnigtcApwGcw 3. **Call the** /api/pokemon_sightings API point using jQuery **$.getJSON** method. a. **Process** the returned sightings data. i. Resolve pokemon's id to pokemon's image (sprite) using the PokemonAPI. ii. Resolve the GPS location to the distance away from your current location. 1. Grab your current location using the HTML5 Geolocation API.

DISTANCE AWAY
13 MILES

DISTANCE AWAY
50 MILES

DISTANCE AWAY
83 MILES

DISTANCE AWAY
18 MILES

DISTANCE AWAY
69 MILES

DISTANCE AWAY
74 MILES

2. Clean the GPS string returned by the pokemon_sightings API by removing the @ prefix and z marker (e.g., @30.2705109,-97.7528052~~,18.98z~~")
3. Use the Google Maps JS API to obtain the distance from the pokemon to you.

b. Populate the resolved data into the table provided using a templating strategy.

i. Select the template row from the DOM.
ii. Clone the template and remove the template class using jquery's **removeClass.**
iii. Modify the template_clone with the resolved data.
iv. Inject the template_clone into the table.

c. For submission,

i. Set your GPS coordinates using the Chrome > Console > Sensor tool to the following:
   1. 32.733353,-97.1094854

[Screenshot of Pokemon Sightings from position 32.733353,-97.1094854]

CONCEPT INTERACTION (30 PTS)

Considering the APIs you have used in this assignment, present a concept interaction that:

- maximizes the spatial and encyclopedic affordances of the computer
- communicate not only the look and feel, but also the role of the interaction.

Demonstrate and document your design process in a FIGMA.

[https://www.figma.com/file/2ub35Ea5JTzdOijU4YYyo7/Module-4-IR?node-id=1%3A5]

Check ++ Criteria:
Craftsmanship
Iteration & Critique
Interactive FIGMA prototype

**** MAKE SURE THIS DOCUMENT HAS OPEN SHARING PERMISSIONS BEFORE TURNING IN.****