Week 5 Lecture 13

Theory

What's in this lecture?

- Functional Programming in JavaScript with Underscore.JS

Functional Programming

- In Functional Programming, strive to separate programs into mostly *pure* and fewer impure functions

- *pure* functions compute results based solely on their inputs: that is, they have no side-effects

- Impure functions may cause side effects, such as assigning to a global variable, logging to console, or calling alert()

Pure Functions

```
function f(x, y) { return x + y; }

function min(a, b) {
  if (a < b) {
    return a;
  }
  return b;
}

function contains(tofind, astring) {
  if (astring.indexOf(tofind) != -1) {
    return true;
  }
  return false;
}
```

Impure Functions

```
function do_stuff(a, b) {
  alert("got:" + a);  // raises a dialog box on-screen

  return a + b;
}

var x = 0; // global scope

function increment_x() {
  x = x + 1; // mutates global scope
}

function get_a_rand() {
  // nondeterministic - returns different values
  return Math.random() * 10;
}
```

Underscore JS

- Provides useful functions for functional programming in JavaScript
- May be used by including underscore.js in a <script> tag in HTML
- Or, by downloading underscore.js and using _ = require("/path/to/underscore.js"); from nodejs.

Map

- Turns a list of input elements into a list of output elements using a given function
- First argument: a list of “x” elements to translate to “y” elements
- Second argument: a *pure* function “f” that turns “x” elements to “y” elements, that is, f(x) = y

Map

```
function plus_one(x) {
  return x + 1;
}

val x = _.map([1, 2, 3], plus_one); // [2, 3, 4]

function get_value(v) {
  return v["value"];
}

val y = _.map([{"value":3}], get_value); // [3]
```

Reduce

- Also known as *fold* or *fold-left*
- Processes a list of "x" values and accumulates them into an "a" accumulator
- For example, "sum", or "word count"
- First argument: a list of "x"
- Second argument: a function of "x" and "a" that returns a new "a"
- Third argument: an initial "a"

Reduce

```
function word_count(x, a) {
  return x.split(“ “).length + a;
}

var c = _.reduce([“four score”,“and”,“seven
years ago”], word_count, 0); // 6
```

Other Functions

- *filter* : returns a new list of elements that match a filter function
- *detect* : uses a function to find the first element in the list that passes a filter function
- *head*, *tail* : use similar to car and cdr in scheme

Exercises

- Read Underscore.JS documentation at http://documentcloud.github.com/underscore/
- Use head and tail in JavaScript to implement a recursive function over a list of elements
- Use map() to turn a list of AJAX results (such as JSON returned from Twitter API) into a list of transformed objects (subset of data)
- Use reduce() to accumulate AJAX results (such as JSON returned from Twitter API) into a "table of contents" object that describes the results
- Use reduce() to accumulate results into a specified DOM element by id (for example, fold them into a <UL> user list element)