Week 1 Lecture 3

Applied

Getting Ready

- Be comfortable with HTML, CSS, and poking around FireBug
- Play around with a few JavaScript functions

What’s in this lecture?

- Functional JavaScript & JQuery

Higher Order Functions

```
function driver(f, x, y) {  // f is passed as an argument
  return f(x, y);            // ... and used as a function!
}

function sum_squares(x, y) {
  return (x * x) + (y * y);
}

function diff_cubes(x, y) {
  return (x * x * x) - (y * y * y);
}

driver(sum_squares, 4, 3);  // 25
driver(diff_cubes, 5, 3);  // 98
```

Sum-Of-Terms Driver

```
function sum_terms(term, a, next, b) {
  if (a > b) {
    return 0;
  }

  return term(a) + sum_terms(term, next(a), next, b);
}
```

Sum-Of-Terms (Tail-Rec)

```
function sum_terms(term, a, next, b) {
  return sum_terms_iter(0, term, a, next, b);
}

function sum_terms_iter(accum, term, a, next, b) {
  if (a > b) {
    return accum;
  }
  return
    sum_terms_iter(
      accum + term(a), term, next(a), next, b);
}
```

Adding Filter

```
function sum_terms(term, a, next, b, filter) {
  return sum_terms_iter(0, term, a, next, b, filter);
}

function sum_terms_iter(accum, term, a, next, b, filter) {
  if (a > b) { return accum; }

  var ta = 0;
  if (filter(a)) {
    ta = term(a);
  }
  return sum_terms_iter(
      accum + ta, term, next(a), next, b, filter);
}
```

Using Filter

```
function identity(x) {     return x;  }

function inc(x) {           return x + 1;  }

function odd_filter(x) { return (x % 2) == 1; }

function even_filter(x) {
  return (x % 2) == 0;
}

sum_terms(identity, 1, inc, 10, odd_filter); // 25
sum_terms(identity, 1, inc, 10, even_filter); // 30
```

Lambda 1

```
// usual way of defining a function
function plus_one(x) { return 1 + x; }

// uses a lambda (closure)
var plus_one = function(x) { return 1 + x; }

// f(a, b): returns f(x, y) = ax + by^2
function axby2(a, b) {
  return function(x, y) {
    return (a * x) + (b * y * y);
  };
}
```

Lambda | Cont’d

```
//
// f(a, b): returns f(x, y) = ax + by^2
//
function axby2(a, b) {
  return function(x, y) {
    return (a * x) + (b * y * y);
  };
}

var a7b92 = axby2(7, 9);  // a7b92 is a *function*

a7b92(3, 4);
```

Lambda 2

```
function chain(f, g) {
  return function() {
    if (f()) {
      return true;
    } else {
      return g();
    }
  };
}

function not_true() { return function() { return false; }; }
function not_false() { return function() { return true; }; }

chain(not_true, not_true)();
chain(not_false, not_false)();
```

Calculator (HTML)

```
<html>...<body><script>...</script>
<form onsubmit="return false;">
<input type="text" id="v1" /><br />
<input type="text" id="v2" /><br />
<select id="op" />
  <option value="plus">plus</option>
  <option value="times">times</option>
</select><br />
<input type="text" id="result" /><br />
<input type="submit"
  value="Do it!" onclick="calc();"/>
</form></body></html>
```

Calculator (JS)

```
function calc() {
  var oper = $("#op")[0].value;
  var operFun = getFun(oper);

  var value1 = parseInt( $("#v1")[0].value );
  var value2 = parseInt( $("#v2")[0].value );

  $("#result")[0].value = operFun(value1, value2);
}

function getFun(n) {
  if (n == "plus") {
    return function(x, y) { return (x + y); }
  } else if (n == "times") {
    return function(x, y) { return (x * y); }
  } else { ... }
}
```

Exercises

- Extend the Calculator with minus, div, and one other custom function
- Add a JavaScript form to one of your web pages that does something interesting (like calculator but different)
- Implement the boat game in JavaScript (note: it's different than in scheme)