Week 5 Lecture 15

Theory

What's in this lecture?

- Inheritance and Object-Oriented Programming in JavaScript

Inheritance

- With *inheritance*, we create sub-classes that behave like their parent class (or superclass)
- The key property is that a subclass is always substitutable for the superclass
- For example, a Dog class may contain eat() and goToKennel() methods; the Labrador and Wolf sub-classes might choose to override those methods

OOP Dogs

```
function Dog(name, food) {
  this.name = name;
  this.food = food;
}

Dog.prototype = {
  eat : function(meal) {
    if (meal === this.food) {
      console.log("Yum! I like " + this.food);
    } else {
      console.log("What is this " + meal + " you give me?");
    }
  },
  greet : function() {
    console.log("hello, my name is " + this.name);
  }
};
```

Creating Subclasses

```
function Labrador(name) {
  Dog.call(this, name, "puppy chow");
}

Labrador.prototype = new Dog();
Labrador.prototype.constructor = Labrador;

var l = new Labrador("fido");
l.greet();
l.eat("grass");
l.eat("puppy chow");
```

Creating Subclasses

```
function Wolf(name) {
  Dog.call(this, name, "rabbits");
}

Wolf.prototype = new Dog();
Wolf.prototype.constructor = Wolf;

var w = new Wolf("danger");
w.greet();
w.eat("grass");
w.eat("rabbits");
```

Another OOP Example

- Shape is super class, has circumference() and area()
- NOTE: in this example, we override all methods - usually, the superclass holds common functionality

- Square is subclass: circumference is 4*side, area is side*side
- Circle is subclass: area is PI*radius*radius, circumference is 2*PI*radius
- Rectangle is subclass: area is length*width, circumference is 2*length+2*width

OOP Shapes

```
function Shape() {}

Shape.prototype = {
  circumference : function() {
    alert("I don't know how to do that");
  },
  area : function() {
    alert("I don't know how to do that");
  }
};
```

OOP Shapes

```
function Square(side) {
  Shape.call(this);
  this.side = side;
}

Square.prototype = new Shape();
Square.prototype.constructor = Square;

Square.prototype.circumference =
  function() { return 4 * this.side; };

Square.prototype.area =
  function() { return this.side * this.side; };

var s = new Square(3);
```

OOP Shapes

```
function Circle(radius) {
  Shape.call(this);
  this.radius = radius;
}

Circle.prototype = new Shape();
Circle.prototype.constructor = Circle;

Circle.prototype.circumference =
  function() { return 2 * this.radius * Math.PI; };

Circle.prototype.area =
  function() { return Math.PI * this.radius * this.radius; };

var c = new Circle(3);
```

Exercises

- Implement Rectangle, Oval and Trapezoid shapes
- Create a Collection class that List and Binary Search Tree inherit from; it should have 2 methods: size() (which gives the size of the collection) and toArray() (which copies all values into a new array and returns it)
- Using inheritance, create an object model to simulate a parking lot, including ParkingLot, ParkingSpace, Vehicle, Motorcycle, SemiTrailer and Car classes; the ParkingLot instance should support a parkVehicle(vehicle) method, refuse vehicles that don't fit, and be able to say how many spaces are left