# Object Modeling Recap

Creating models out of real-world concepts

# Today’s Outline

- Recap of Modeling Objects and OO
- Group Modeling Exercises of a real-world concept
- Open Questions
- Workshop - getting comfortable at the keyboard

# Goals

- Understand how / why we organize data into highly specific data structures.
- Recognize how an object is created.
- Get more familiar writing javascript - most important for newer coders.

# Modeling Objects - Why

Without using an object, given a first name, a last name and an email, how do we store objects in memory in javascript?

```
var firstName = ‘Noob’;

var lastName = ‘Saibot’;

var email = ‘noobsaibot@gmail.com’;
```

# Modeling Objects - Why

Storing multiple variable names quickly becomes fragile and leads to messy, uncoordinated code.

Modeling objects is the craft of putting together pieces that best describe a real-world concept and leaving out parts that belong elsewhere.

We can better *model* a person in code using full-fledged objects.

In javascript there are a few ways to encapsulate the properties of an object.

# Modeling Objects - How

The object-literal pattern should be your very first instinct on how to create an object in javascript

Given the same necessary properties (first and last name, email), how do we create an object using object-literal?

# Modeling Objects - How

As a refresher, objects are created using the object-literal pattern by using opening and closing curly-braces.

```
var myObject = { };
```

Add *properties* to an object-literal on creation by using the ‘ : ’.

```
var myObject = {
        propertyName: ‘propertyValue’,
        numericName: 11
};
```

# Modeling Objects - How

After you create an object, since javascript is a dynamic language, you can add new properties whenever you want.

Add or change the value of a property using the ‘ . ’ (dot operator).

```
var myObject = { };

myObject.propertyName = ‘Property Name’;
myObject.propertyName = ‘Changing the name here.’;
```

# Modeling Objects - How

Remember that objects can have any amount or type of properties you want assigned to them.

A property of an object can be a number, string, function or an array.

# Modeling Objects - How

```
var course = {
        name: ‘US History 101’,
        instructor: {
                name: ‘George Burns’
        },
        numberOfStudents: 30,
        requiredTextBooks: [
                ‘New York: The Early Days’,
                ‘Presidents A - Z’
        ]
};
```

# Modeling Objects - How

Still using the object-literal method, how do we create an object to represent an article written on buzzfeed?

# Modeling Objects - How

Once we have objects modeled out, we can begin to add behavior.

Objects begin to get more interesting once we start adding methods.

Methods are simply functions that belong to objects.

We can add functions to an object-literal in the same manner as any other property.

# Modeling Objects - How

```
var car = {
        type: ‘Oldsmobile’,
        currentSpeed: 0,
        maxSpeed: 55,
        drive: function () {
                console.log(‘vroom, going: ’ + this.currentSpeed + ‘ mph’);
        }
};
```

# Modeling Objects: How

The complexity begins when you have more than one object or idea you need to model.

You will begin to ask yourself what object should be responsible for what behavior.

This is why this process can be thought-intensive and time-consuming. Good models save you lots of time.

# Modeling Objects: How

Using the previous “car” example, let’s say we wanted to add a driver?

What are the driver’s responsibilities and how do we model the driver?

We can begin by outlining the basic properties of a driver.

# Modeling Objects: how

```
var driver = {
        name: ‘Kyle Pace’,
        hasTakenDriversEd: true,
        hasInsurance: true,
        yearsDriving: 16
};
```

How do we make this driver drive a car?

# Modeling Objects: How

Give the driver a car

```
var driver = {
        name: ‘Kyle Pace’,
        hasTakenDriversEd: true,
        hasInsurance: true,
        yearsDriving: 16,
        car: car,
        drive: function () {
                car.drive();
        }
};
```

# Modeling Objects: How

In other circumstances, it is perfectly reasonable to assign the driver to the car instead of how we ended up modeling the domain.

These are the choices that require thought into how your system might expand.

Proper modeling requires an intimate understanding of the domain you are working within.

# Questions?

Let’s take a screen from a popular application and model out the properties and behavior.  Any suggestions?

Before moving on, any questions regarding *any* material covered at the class thus far?

# Workshop

- Get Comfortable Coding!
- If you are nervous or unsure about typing code at the keyboard, it’s your turn tonight.
- Use this workshop to get more familiar with the vocabulary we are taking for granted.
- The exercises are designed to be finished in a few lines of code but are written without too many layman's terms, consult the nearest coder for translation but *write the code yourself*.