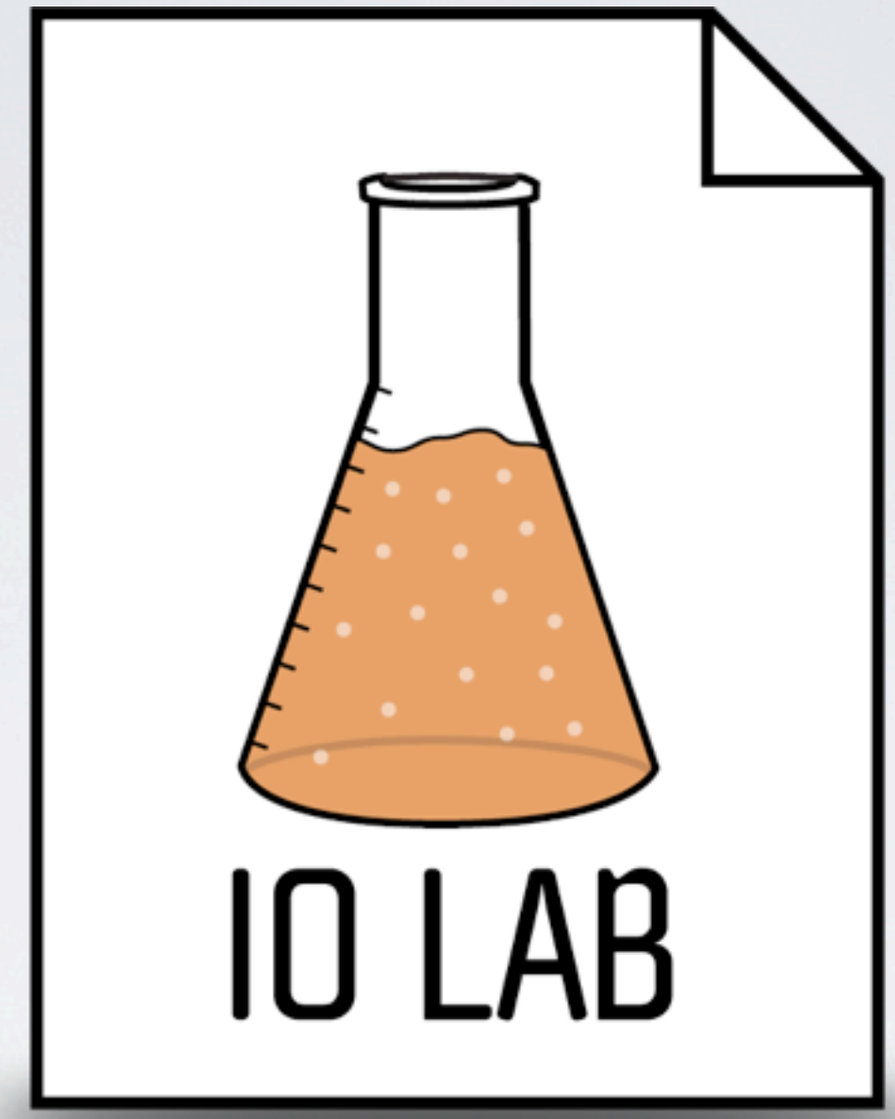

INFORMATION ORGANIZATION LAB

LAST TIME ON IOLAB

TEST DRIVEN DEVELOPMENT

TODAY

OBJECT ORIENTED JAVASCRIPT

WHAT IS IT?

Object-oriented programming (**OOP**)
a programming paradigm using "objects" consisting of "properties" and "methods" together with their interactions - to design computer programs.

Properties
hold values associated with the object

Methods
provide definitions of how the object can be acted on

Source: http://en.wikipedia.org/wiki/Object-oriented_programming

WHY SHOULD WE CARE?

- Modularity (eliminate task duplication & reuse code)
- Abstraction (trust the implementation)
- Encapsulation (isolate variables/functions from other code)

EXAMPLES

Native Javascript Objects

Date Object

```
var d = new Date(); //instantiate Date object

d.getDay(); //method that returns day of week
```

EXAMPLES

Native Javascript Objects

Array Object

```
var arr = new Array(); //instantiate
arr[0] = “foo”;
arr[1] = “bar”;
arr[2] = “baz”;

arr.length; // property (returns 3)
arr.indexOf(“foo”); // method (returns 0)
```

Above is just an example, preferred way is still:

```
var arr = [“foo”, “foo”, “foo”];
```

CREATING OBJECTS

Defining a "class"

```
function Foo(){} // yup, that's it.

var f = new Foo();
var f2 = Foo();

typeof f // ???
typeof f2 // ???
```

CREATING OBJECTS

Defining a "class"

```
function Foo(){} // yup, that's it.

var f = new Foo(); // create instance of Foo obj
var f2 = Foo(); // assign the function Foo to f2

typeof f // object
typeof f2 // function
```

CREATING OBJECTS

Dynamically adding properties

```
function Foo(){}

var f = new Foo();
var f2 = new Foo();
f.x = 1;
f.y = 2;

console.log(f.x); // 1
console.log(f.y); // 2
console.log(f2.x); // ??
console.log(f2.y); // ??
```

CREATING OBJECTS

Defining a "class" - Constructors

```
function Foo(){ // constructor
  this.x = 1; // property
  this.y = 2;
}

var f = new Foo();
var f2 = new Foo();

console.log(f.x); // 1
console.log(f.y); // 2
console.log(f2.x); // ??
console.log(f2.y); // ??
```

PROTOTYPE

JavaScript is a prototypal language

All objects have a base prototype.

Prototypes serve as "templates" or "blueprints" for an entire class.

Prototype properties are shared by all instantiated objects of the class.

CREATING OBJECTS

Defining a "class" - Methods

```
function Foo(){ // constructor
   this.x = 2; // property
   this.y = 3;
}

Foo.prototype.z = function () { // method
   return 4;
}

var f = new Foo();
f.z; // 4;
```

CREATING OBJECTS

Defining a "class" - Methods

```
function Foo(){ // constructor
  this.x = 2; //property
  this.y = 3;
}

Foo.prototype.multiply = function () { //method
  return this.x * this.y;
}

var f = new Foo();
f.multiply(); // 6;
```

CONTROLLING ACCESS

```
function Foo(){
  this.x = 2;
  function subtract (){
    return this.x - 1;
  }
  this.add = function() {
    return this.x - 1;
  }
}

Foo.prototype.double = function () {
  return this.x * 2;
}

var f = new Foo();
f.double(); // ???
f.add(); // ???
f.subtract(); // ???
```

CONTROLLING ACCESS

```
function Foo(){ // constructor
  this.x = 2;
  function subtract (){ //private
    return this.x - 1;
  }
  this.add = function() { //privileged
    return this.x - 1;
  }
}

Foo.prototype.double = function () { //public
  return this.x * 2;
}

var f = new Foo();
f.double(); // 4
f.add(); // 3
f.subtract(); // ERROR!!
```

INHERITANCE

Creating a specialized version of a class

Parent / super-class
Original class

Child / sub-class
Specialized class w/ inherited methods & properties from parent

DEFINING A SUB-CLASS

```
function Animal(name){ // parent constructor
  this.name = name;
}
Animal.prototype.sayName = function () {
  console.log(this.name);
}

function Dog(name){ // child constructor
  Animal.call(this, name);
  this.collarText;
}
Dog.prototype = new Animal();
Dog.prototype.setCollarText = function(text){
  this.collarText = text;
}

var d = new Dog("Fido");
d.sayName(); // Fido (inherited method)
d.setCollarText("FIDO"); // sub-class method
```

POLYMORPHISM

```
function Animal(name){ // parent constructor
  this.name = name;
}
Animal.prototype.speak = function () {
  console.log(this.name + " says: ");
}

function Dog(name){ // child constructor
  Animal.call(this, name);
}
Dog.prototype = new Animal();

Dog.prototype.speak = function(){ // override parent method
  Animal.prototype.speak.call(this); // call parent method
  console.log("woof woof!");
}

var d = new Dog("Fido");
d.speak(); // Fido says: woof woof!
```

OOP EXAMPLE

To-do List

https://github.com/iolab12/iolab_hw_1

OPEN LAB

Project 3

FOR NEXT TIME

Python Scripting

Reminder: Project 3 due 11/14

You can find links to help with all of these on the course website at
http://courses.ischool.berkeley.edu/290ta-iol/f12