Java Script Training

Exercises

You will be provided a zip file with the scaffolding for the exercises.

The exercises are supposed to be completed using plain JavaScript (without 3rd party libraries, except for the promise based API)

Once the exercises are complete, re-zip the files as they were provided and send them by email to the instructor.

Due date

October 20th, 2014

Honor code

- I will provide creative code
- My answers to the exercises will be my own work
- I will not make solutions to the exercises available to anyone else.
- I will not engage in any other activities that will dishonestly improve my results or dishonestly improve/hurt the results of others.

Functions and Modules

Create the *utils* module with the following functions

- extend
- bind
- memorize
- lazy

Note: implement in **js/utils.js**

utils#extend

Copy all of the properties in the source objects over to the destination object, and return the destination object.

```
utils.extend({name : 'moe'}, {age : 50});
=> {name : 'moe', age : 50}
```

utils#bind

Bind a function to an object, meaning that whenever the function is called, the value of *this* will be the object (Note, implement without using native bind).

```
var func = function(greeting){
  return greeting + ': ' + this.name;
};
func = utils.bind(func, {name : 'Matias'}, 'hi');
// Expected behavior
func();
=> 'hi: Matias'
```

utils#memoize

Memoize a given function by caching the computed result.

```
var fibonacci = function(n) {
  console.log('Fib ' + n);
  return n < 2 ? n : fibonacci(n - 1) + fibonacci(n - 2);
}
var memoizedFibonacci = utils.memoize(fibonacci);
// Expected behavior
memoizedFibonacci(5);
// 1st time, I should see "Fib 5",..,"Fib 0"
memoizedFibonacci(5);
// 2nd time, I shouldn't see any "Fib x" because the result should be cached
```

utils#lazy

Create and return a new lazy version of the passed function that will postpone its execution until after wait milliseconds have elapsed since the last time it was invoked.

```
// e.g. writing to localStorage is a sync operation
var logger = function(msg) {
  console.log(msg);
}
var lazyLogger = utils.lazy(logger, 1000);
// Expected behavior
lazyLogger(someMsg);
// after 500ms I should not see any log message
lazyLogger(someOtherMsg);
// after 1001ms I should see someOtherMsg
// Notice that lazyLogger should lazily call logger with the last supplied
arguments
```

Inheritance, Mixins, OLOO

Implement the following pseudo-code with 2 approaches

- Classical Inheritance
  Combining classical patterns (ie Constructor Stealing and Pseudo-classical)
  Note: implement in **js/vehicle-classical.js**
- Mixins or OLOO
  Try to be creative and think what would be the easiest way to use your API
  Note: implement in **js/vehicle-mixins.js**

For the 2nd approach remember

JavaScript, being a loosely typed language, never casts. The lineage of an object is irrelevant. What matters about an object is what it can do, not what it is descended from.

```
class Vehicle {
  engines = 1
  ignition() {
    output( "Turning on my engine." );
  }
  drive() {
    ignition();
    output( "Steering and moving forward!" )
  }
}

class Car inherits Vehicle {
  wheels = 4
  drive() {
    inherited:drive()
    output( "Rolling on all ", wheels, " wheels!" )
  }
}

class SpeedBoat inherits Vehicle {
  engines = 2
  ignition() {
    output( "Turning on my ", engines, " engines." )
  }
  pilot() {
    inherited:drive()
    output( "Speeding through the water with ease!" )
  }
}
```

Callback and Promises

Convert the provided callback based API to a promise based API (using Q.js) so that it can be consumed like

```
// Promise based API (see js/asyn-promise.js)
// Note: async-promise.html has already q.js included

Programs.get()
  .then(function(programs) {
    console.log('Received', programs);
  });
Programs.schedule('someProgramId')
  .then(function(scheduled) {
    console.log('Schedule result', scheduled);
  });
```

Note: implement in **js/async-promise.js**

```
// Callback based API (see js/async-callback.js)

Programs.get(function(movies) {
  console.log('Received', movies);
});
Programs.schedule('someMovieId', function(scheduled) {
  console.log('Schedule result', scheduled);
});
```