Javascript

Introductions

Class Overview

- Language Basics
- TDD with Jasmine
- Browser DOM, CSS
- Asynchronous requests (AJAX)
- JQuery overview

HISTORY

- JavaScript was invented in the 90's to make webpages more interactive
- Netscape named their implementation "JavaScript" after the Java language because it was popular at the time
- IE came out with a similar implementation called JScript
- The ECMAScript standard was created to standardize the language

- Firefox
- Firebug
- Web server (apache)
- Aptana

Language Basics

- Syntax similar to Java
- JavaScript is loosely typed. Variables are declared with a generic “var” rather than String, int, float, etc.
- The basic types: undefined, null, boolean, string, number, object

- Number type

```
var myNumber1 = 1;
var myNumber2 = 2;
var mySum = myNumber1 + myNumber2;
console.log('mySum: ', mySum);
```

- String type

```
var myString1 = "string 1";
var myString2 = "string 2";
var myConcat = myString1 + myString2;
console.log('myConcat: ', myConcat);
```

```
var myCombo1 = myString1 + myNumber1;
console.log('myCombo1: ', myCombo1);

var myCombo2 = myNumber1 + myString1;
console.log('myCombo2: ', myCombo2);
```

- Object == hash == associative array
- You can create an object like

```
var myObject = new Object();
```

- OR

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

- Object is the superclass of all objects

- Null

```
var nullVar = null;
console.log('nullVal: ', nullVar);
console.log('is nullVar null? ', nullVar == null);
console.log('is nullVar undefined? ', nullVar === undefined);
```

- Undefined

```
var myObj = new Object();
console.log('undefinedVar: ', myObj['undefinedVar']);
console.log('is undefinedVar null? ', myObj['undefinedVar'] ==
null);
console.log('is undefinedVar undefined? ', myObj['undefinedVar']
=== undefined);
```

- Things that are false:
  - false, 0, ‘’, undefined, null

```
if (myObj['undefinedVar']) {
    console.log('undefinedVar is true');
} else {
    console.log('undefinedVar is false');
}
```

- Everything else is true

- Arrays
- Create an array like:

```
var myArray = [1,2];
console.log('myArray: ', myArray);
```

- Or like:

```
var myArray2 = new Array();
myArray2.push(1);
myArray2.push(2);
console.log('myArray2: ', myArray2);
```

- toString()

```
console.log('myArray2 toString: ', myArray2.toString());
```

- Quotation

- “ or “”
- “ most common

- functions

```
function sayHello(name) {
  console.log('hello ' + name);
}

sayHello('lorien');
```

- Can be global or a member of an object

Objects

- Object == hash == associative array
- You can create an object like

```
var myObject = new Object();
```

- OR

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

- Object is the superclass of all objects

- You can set properties on an object like

```
myObject.a = "a";
```

- OR

```
myObject['a'] = "a";
```

- You can access the properties of an object like

```
myObject.a;
```

- OR

```
myObject['a'];
```

- You can set a function on an object like

```
myObject.myFunction = function() {
    console.log('function prints myObject.a: ', myObject.a);
};
```

- And you can call a function like

```
myObject.myFunction();
```

- You can see all of the attributes and functions of an object like

```
for (var prop in myObject) {
    console.log(prop, ": ", myObject[prop]);
}
```

Iterating

- Iterate over an array using an index:

```
var myArray = ['a', 'b', 'c', 'd'];

for (var i=0; i<myArray.length; i++) {
    console.log(myArray[i]);
}
```

- Iterate over a hash with for in

```
var myHash = new Object();
myHash['a'] = 'a';
myHash['b'] = 'b';
myHash['c'] = 'c';
myHash['d'] = 'd';

for (var key in myHash) {
    console.log("key: ", key, ", value: ", myHash[key]);
}
```

- Don’t iterate over an array as if it were a hash

```
var myArray = ['a', 'b', 'c', 'd'];
myArray['contains'] = function(){};
for (var key in myArray) {
    console.log('key: ', key, ', value: ', myArray[key]);
}
```

- Iterate using index so you don’t get extra properties

```
for (var i=0; i<myArray.length; i++) {
    console.log(myArray[i]);
}
```

Classes

- A class is a “blueprint” that describes the attributes and behaviors of a type of object.

```
function Car() {

  this.numDoors = 2;
  this.color = 'red';
  this.isDriving = false;

  this.drive = function(){
    this.isDriving = true;
  }

  this.park = function(){
    this.isDriving = false;
  }

}
```

- You can create a new instance of the Car class like

```
var camero = new Car();
camero.color = "Electric Blue";
```

- If you print the value of isDriving it will evaluate to false. To drive the car call

```
camero.drive();
```

- Now isDriving evaluates to true

- You could also define the properties of a class by modifying its “prototype”

```
function Car3() {};

Car3.prototype.numDoors = 2;
Car3.prototype.color = 'red';
Car3.prototype.isDriving = false;

Car3.prototype.drive = function(){
  this.isDriving = true;
};

Car3.prototype.park =  function(){
  this.isDriving = false;
};
```

- Augmenting a class by modifying its prototype

```
Array.prototype.contains = function(item) {
  //determine if the array contains the item
};
```

Determining Type

- Determining type with ‘instance of’

```
console.log('camero instance of Car? ', camero instanceof Car);
```

- Determining type with ‘typeof’

```
console.log('typeof myCar: ', typeof myCar);

console.log('typeof myCar.drive: ', typeof
myCar.drive);
```

Activity

- Shopping Cart
- Item class
- ShoppingCart class

Homework

- Fill in implementation for empty methods in ShoppingCart class.
- Modify MagicalItem and ShoppingCart classes to use prototype syntax.
- Read Section 4 Overview, and Section 8 types (only 8.1-8.5) in the EcmaScript specification: http://www.ecma-international.org/publications/standards/ecma-262.htm

Shopping Cart Class

```
function ShoppingCart() {

    this.items = new Array();

    this.addItem = function() {
        //add the item to the array
    }

    this.getTotalCost = function() {
        //iterate over array and sum up the prices
    }
}
```