Objects in JavaScript

Lecture 5
Chapter 6 in JavaScript text

Note on the textbook

If you are keeping up with the reading, you see that the JavaScript book is **very** detailed.

Chapter 6 introduces many advanced topics regarding classes and objects

You need to read the entire chapter. I'm covering the most commonly used parts here - but we will eventually all of it.

Objects in JavaScript

Objects in JavaScript are a very big departure from what you’ve learned from C++ and Java

Objects are best thought of as **collections** of named values (properties) - of any type.

Unlike C++ however - an object’s ***class*** doesn’t need to be defined - you can add and remove properties at will!

Objects in JavaScript

- An object is an unordered list of properties and associated data.
  - This type of structure is often called an *associative array*.
- Property name are strings
  - Its a lot like a Map - where the key is a string.

```
var me = new Object();
me.firstName = “Scott”
console.log(me.firstName)
console.log(me[“firstName”])
```

Creating an object

There are 3 ways to create and object:

```
1. var me = { };
2. var me = new Object();
3. var me = Object.create(null);
```

Lets deal with them 1 at a time

JSON

Unlike other languages, in JavaScript it is very common to have object **literals**.

```
var x = 5;   // 5 is an integer literal
var me =  {  // this is an object literal
   “first” : “Scott” ,
   “last” : “Frees”
};
```

Object literals are written in **J**ava**S**cript **O**bject **N**otation (pronounced “jay-s-on”)

JSON

- We’ll add to our understanding of JSON as we go - but for now
  - An object starts with {
  - followed by 1 or more **name : value** pairs separated by commas.
  - value can be any primitive type
  - value can be another object
  - and ending with an }

An empty object is written as { }

Constructors

- A second way to create a new object is to call a constructor
- We'll talk more about this later - its not quite like C++'s concept of constructors.

There are a few "built in" objects

```
Var o = new Object();
var a = new Array() ; // chapter 7
var d = new Date();
var r = new RegExp();
```

The Object.create() function

The third was is by calling a special function on the object, called *create.*

When we talk about classes and prototypes, we'll see a bit more about this.

```
var me = Object.create(first:"Scott", last:"Frees");
```

Working with properties

```
var me = new Object();
me.firstName = “Scott”
console.log(me.firstName)
console.log(me[“firstName”])
```

Properties can be added to an object at anytime. All objects are of type “Object” - an object’s properties **don’t** make its type!

Working with properties

```
var me = new Object();
me.firstName = “Scott”
console.log(me.firstName)
console.log(me[“firstName”])
```

- You can access properties using the . notation, or as an array - where the property name is a string index.
- This is particularly helpful if the property name is the result of a computation, or is a parameter

Working with properties

Just like regular variables, if you try to **read** a property that doesn't exist, you'll get an error.

```
var me = { };
console.log(me.first);
```

Also, remember that you must initialize the object!

```
var me;
me.crash = “now”;
```

Removing properties

```
var me = { first : “Scott” , last : “Frees” };
delete me.first;
```

You can delete a property that doesn’t exist… nothing happens.

Don’t confuse delete in JavaScript with delete in C++ - it has nothing to do with memory management!

Testing for a property

You can test if a property exists in an object

```
var me = { first : “Scott”, last : “Frees” }
if ( “first” in me ) {
    console.log(“I have a first name”);
}
if ( ! (“middle” in me ) {
   console.log(“I do not have a middle name”);
}
```

Iterating over properties

```
var me = { first : "Scott", last : "Frees" }
var jfk = { first : "John", middle : "Fitzgerald",
             last : "Kennedy" };
var pele = { first : "Pelé"}
for ( name in me ) {
    console.log(me[name]);
}
for ( name in jfk ) {
    console.log(jfk[name]);
}
for ( name in pele ) {
    console.log(pele[name]);
}
```

Note that me.name won’t work - because the intepreter would look for a property called “name”.

Nested objects

```
var daily_diet = {
    breakfast : {
        eat : “eggs”, drink : “coffee”
    },
    lunch : {
        eat : “ham and cheese”, drink : “coffee”
    },
    dinner {
        eat : “steak”, drink : “wine”, dessert : “ice cream”
    }
}
```

Serialization

We often need to convert objects to and from string representations.

```
var me = { first : “Scott” , last : “Frees” };
var s = JSON.stringify (me);
var twin = JSON.parse (s);
```

JSON.stringify is wonderful for debugging

Where are the methods?

- We are used to object having member functions in C++ and Java

- We will have object methods… but we haven't looked at functions in general yet… so we'll come back to this.

Next

Next we will look at Chapter 7 in the JavaScript text - Arrays.

Arrays are special types of objects, which maintain an ordering of properties.

- Each element has a numeric position
- Each element can be any type though
- Arrays are of dynamic size
- Arrays have many useful functions built in