Introduction to JavaScript – Part II

Hamid Zarrabi-Zadeh

Web Programming – Fall 2013
Outline

- JavaScript Objects
- Object Operators
- Constructor Functions
- Data Encapsulation
- Built-in Objects
Objects

- JavaScript objects are collections of name/value pairs
- Objects in JavaScript are similar to
  - Dictionaries in Python
  - Hashes in Perl and Ruby
  - Associative arrays in PHP
  - HashMaps in Java
Object Structure

• The **name** part is a simple string, while the **value** can be any JavaScript value, including other objects

```javascript
var person = {
    name: 'Hamid',
    family: 'Fereydoon',
    id: 12
}
```
Object Properties

- The values in an object are usually called properties
- Property names can also be numbers

```javascript
var person = {
  1: 'Hamid',
  2: 'Fereydoon',
  others: 0
}
```
Creating Objects

• Two ways to create objects
  
  ```javascript
  var obj = {};
  var obj = new Object();
  ```

• These two are equivalent
Accessing Properties

- Object properties can be accessed through the . or [] operators

```javascript
var person = {
  name: 'Hamid',
  11: '9121122090'
}

person.name
person['name']
person[11]
person[name]     // Error: name is not defined
```
Nested Objects

```javascript
var obj = {
    name: 'T-Shirt',
    'for': 'ACM',
    details: {
        color: 'Black',
        size: 10
    }
};

obj.details.color; // Black
obj['details']['size']; // 10
```
Object Operators
Update

- We can add/update properties on the fly, using access operators

```javascript
var obj = {
    name = 'Ali'
};

obj.name = 'Hamid';
obj['family'] = 'Fereydoon';
```
Delete

• We can remove properties using `delete` operator
• Can be used to remove objects (as they are properties of global `window` object)

```javascript
var obj = {
    name: 'Ali'
};

delete obj.name
delete obj[1]
delete obj
```
Property Existence

- The `in` operator determines whether an object has a certain property

```javascript
if ('property' in obj) { ... }
if (typeof obj.property !== 'undefined') { ... }
if (obj.hasOwnProperty('property')) { ... }
```
The `typeof` operator returns a string representing the type of the argument.

Can be one of:
- "function"
- "string"
- "number"
- "boolean"
- "object"
- "undefined"
Classes and Objects
Constructor Functions

• JavaScript doesn't really have classes!
• Instead, you define a constructor function that sets the properties of the implicit variable `this`

```javascript
function Person(first, last) {
  this.first = first;
  this.last = last;
  this.fullName = function() {
    return this.first + ' ' + this.last;
  }
}

var s = new Person('Hamid', 'Rohani');
alert(s.fullName());
```
Check Instances

- The `instanceof` keyword tests if an object is of a specific class
- This really checks if the object was created using the named constructor function

```javascript
var c = new Person('Hamid', 'Rohani');
if (c instanceof Person) {
    alert('c is a Person');
}
```
Prototypes

- Constructor functions have a property named `prototype` that allows for the creation of properties and methods.

```javascript
function Person(first, last) {
    this.first = first;
    this.last = last;
}

Person.prototype.fullName = function() {
    return this.first + ' ' + this.last;
}
```
Prototype Chain

• Person.prototype is an object shared by all instances of Person
• It forms part of a lookup chain called prototype chain
• Any time you attempt to access a property of Person which is not set, JavaScript will check Person.prototype for that property
Modifying Prototypes

• You can modify prototypes at any time, which means you can add extra methods to existing objects at runtime

```javascript
String.prototype.reversed = function() {
    var r = '';
    for (var i = this.length - 1; i >= 0; i--) {
        r += this[i];
    }
    return r;
}
'Hamed Alavi'.reserved();
```
Invoking Functions

- We can invoke a function on an object using `call` and `apply` methods

```javascript
function add(a, b) {
    return a + b;
}

// three equivalent ways to call add
add(2, 3);
add.call(null, 2, 3);
add.apply(null, [2, 3]);
```
Invoking Functions (cont'd)

- The first parameter given to `call` and `apply` methods will be set to the function's internal `this` value

```javascript
function log(x) {
    return this + ': ' + x;
}

log.call('Cat', 'Hello!');  // Cat: Hello!
```
Data Encapsulation
Namespace

• One way to minimize the use of global variables is to create a single global variable (namespace) for your application

```javascript
var MyApp = {}; 
MyApp.size = 10; 
MyApp.Person = { ... }; 
MyApp.f = function() { ... }; 
```
Data Hiding

• Another trick is to hide global variables inside a function

(function(){
    var a = 10;
    var b = 20;
    function f(x) { ... };  
    
    b = f(a);
})()
Built-In Objects
Built-In Objects

• Some basic objects are built-in to JavaScript
  – String
  – Array
  – Date
  – Boolean
  – Math
Strings

• A `String` object is created every time you use a string literal (just like in Java)
• Have many of the same methods as in Java
  – `charAt`, `concat`, `indexOf`, `lastIndexOf`, `match`, `replace`, `search`, `slice`, `split`, `substr`, `substring`, `toLowerCase`, `toUpperCase`
• There are also some HTML specific methods
  – `big`, `blink`, `bold`, `fixed`, `fontcolor`, `fontsize`, `italics`, `link`, `small`, `strike`, `sub`, `sup`
• Don't use the HTML methods (use CSS instead)
Arrays

• An **Array** object can be easily created by enumerating its items

• Properties
  – length

• Methods
  – concat, indexOf, join, lastIndexOf, pop, push, reverse, shift, slice, sort, splice, toString, unshift

```javascript
var a = ['ali', 20, 14];
a.push(10);
a.sort(); // [10, 14, 20, "ali"]
```
Dates

- The **Date** class makes working with dates easier
- Some methods
  - `getYear`, `getMonth`, `getDay`, `getHours`, `getMinutes`, `getSeconds`, `getMilliseconds`, `getTime`, `parse()`

```javascript
var today = new Date();
var deadline = new Date(2013, 10, 20);

if (today < deadline) {
    days = (deadline - today) / (3600 * 24 * 1000);
    alert('You have ' + days + ' days left');
}
```
Math

- The **Math** object encapsulates many commonly-used mathematical functions and constants
- **Math functions**
  - abs, acos, asin, atan, atan2, ceil, cos, exp, floor, log, max, min, pow, random, round, sin, sqrt, tan
- **Math constants**
  - E, LN2, LN10, LOG2E, LOG10E, PI, SQRT1_2, SQRT2

```javascript
Math.sqrt(2);
Math.cos(Math.PI);
```
References

• JavaScript: The Good Parts
  – By Douglas Crockford

• A re-introduction to JavaScript

• W3Schools
  – http://www.w3schools.com/js