Java is a trademark of Sun Microsystems, Inc.

JavaOne℠

Beginning JavaScript™ Programming Language for Java™ Technology Developers

Jason Lee
Sun Microsystems, Inc.

A Brief Introduction

- Senior Java Developer for Sun Microsystems
- GlassFish Administration Console Team
- Mojarra Scales Co-founder
  - JSF Component set utilizing much of the Yahoo! User Interface Library

Why Should I Care About JavaScript?

- Hard to write a modern web app without it
- Server-side JavaScript is possible
- It's a really cool language

Topics

- Language Miscellany
- Debugging
- Variable Scope
- Fun with functions
- Classes
- The Document Object Model
- Browser Events
- Testing
- Server-side Java

Language Miscellany

- Execution is top-down in the file
- Functions are defined using *function* keyword
- Blocks can be delimited using curly braces
- Comments can use // or /* */
- Flow control operators are just like Java

Debugging

- NetBeans 6.5+
- Firebug
  - Firebug Lite
- Safari
- Various IE-only options

Variable Scope

- JavaScript variable scope
  - Follows strange rules which are...
    - ...really confusing. Let's see some code!

Variable Scope, cont.

- Global

```
var numberCars = 3; // global
numberTrees = 15; // global
if (numberTrees > numberCars) {
   var numberRoads = 4; // global
   numberFoo = "bar";
} else {
   var numberLakes = 9; // global
}
```

Variable Scope, cont.

- Local

```
function simpleFunction() {
   var colorCar = 'blue'; // local
   colorTree = 'green'; // global, once this
                //function is called
   if (colorCar != colorTree) {
      var colorRoad = 'grey'; // local in
                // this function after this line
   } else {
      var colorLake = 'aqua'; // local, but
                // be undefined since never run
   }
}
```

Variable Scope Recap

- Global Variables
  - Var declared outside a function w/ or w/o the var keyword
  - Var declared outside a function inside a block, but only if and after that block executes
  - Var declared inside a function w/o var keyword, but only if and after the function is called
- Local Variables
  - Var declared inside a function w/var keyword
  - Var declared in a function inside a block w/var keyword, but only if and after the block executes

Array and Object Declaration

- Arrays
  - `var myArray = [1,2,3,4,5];`
  - `var myArray = new Array(1,2,3,4,5);`
- Objects
  - `var myObject = {foo: "one", bar: 2};`

Putting the fun in functions

- Anonymous functions
  - a function (or a subroutine) defined, and possibly called, without being bound to a name

```
subscribe("fakeEvent",
   function(ev, payload) {
      // Do some work here
);
```

Putting the fun in functions, cont.

- Closures
  - occurs when a function is defined within another function

```
function jediMindTrick(text) {
   var fullText="*waves hands* "+text;
   text2="*waves hands* "+text;
   return function() {
      println (fullText);
   }
}
```

Putting the fun in functions, cont.

- The Module Pattern
  - Truly private variables

```
var childManager = (function(){
   var children = [];
   return {
      addChild: function(child{
         children[child.name] = child; },
      removeChild: function(name) {
         children[name] = undefined; },
      getChild: function(name) {
         return children[name]; }
   }
})();
```

JavaScript Classes

- "Prototype-based" not "class-based"
  - There is no class keyword
  - Existing classes (and their definitions) can be changed with the prototype keyword

JavaScript Classes, cont.

Class-based (Java)	Prototype-based (JavaScript)
Class and instance are distinct entities.	All objects are instances.
Define a class with a class definition; instantiate a class with constructor methods.	Define and create a set of objects with constructor functions.
Create a single object with the new operator.	Same.
Construct an object hierarchy by using class definitions to define subclasses of existing classes.	Construct an object hierarchy by assigning an object as the prototype associated with a constructor function.
Inherit properties by following the class chain.	Inherit properties by following the prototype chain.
Class definition specifies all properties of all instances of a class. Cannot add properties dynamically at run time.	Constructor function or prototype specifies an initial set of properties. Can add or remove properties dynamically to individual objects or to the entire set of objects.

JavaScript Classes cont.

- Methods

```
function MyClass() {
   this.foo = “”;
   this.getFoo = function() {return this.foo;}
}
var myClass = new MyClass();
```

- JSON

```
var myClass = {
   foo: “”,
   getFoo: function() { return this.foo };
}
```

JavaScript Classes cont.

- Singleton using an anonymous function

```
var myClass = new function() {
    this.foo = "";
    this.getFoo = function {
        return this.foo;
    }
}
```

JavaScript Inheritance

- Two approaches
  - Function-based
    - `this.inheritFrom = SuperClass;`
    - `this.inheritFrom();`
  - Prototype-based
    - `SubClass.prototype = new SuperClass();`

The Document Object Model

- The Document Object Model (DOM) at the heart of client-side JavaScript programming
  - Same concept as XML DOM
- Base object is *document*
  - `document.getElementById()`
  - `document.getElementsByName()`
  - `document.cookie`

Browser Events

- blur
- change
- click/dblclick
- focus
- keydown/keyup
- keypress
- load
- mouseover/mouseout
- mousedown/mouseup

Testing

- Several options
  - YUI Test
  - JsUnit (two of them!)
  - RhinoUnit

Testing, cont.

- YUI Test

```
var ModuleTest = new YAHOO.tool.TestCase({
    name: "Module TestCase",
    _should: { error: { } },
    setUp : function () { },
    tearDown : function() { },
    testAddChild : function () {
        childManager.addChild({ name: "Timmy", hobby: "baseball"});
        var child = childManager.getChild("Timmy");
        Assert.areEqual(child.name, "Timmy");
    }
});
YAHOO.util.Event.onDOMReady(function (){
    var logger = new YAHOO.tool.TestLogger("testLogger");
    YAHOO.tool.TestRunner.add(ModuleTest);
    YAHOO.tool.TestRunner.run();
});
```

Server-Side JavaScript

- Easily integrate with Java using JSR-223
- Comes with Java 6
- Really easy:

```
ScriptEngineManager manager = new ScriptEngineManager();
ScriptEngine engine = manager.getEngineByName("js");
engine.eval(someScript);
```

- JSR-223 supports at least 25 languages

Server-Side JavaScript, cont.

- Example 1

```
ScriptEngineManager manager =
   new ScriptEngineManager();
ScriptEngine engine =
manager.getEngineByName("js");
String script = Util.readFile("example1.js");

engine.eval(script);

engine.eval("add(1,2);");
```

Server-Side JavaScript, cont.

- Example 2

```
ScriptEngineManager manager =
   new ScriptEngineManager();
ScriptEngine engine =
manager.getEngineByName("js");
String script = Util.readFile("example2.js");

engine.eval(Util.readFile("example1.js"));

engine.put("param1", Integer.valueOf(100));
engine.put("param2", Integer.valueOf(200));

engine.eval(script);
```

Server-Side JavaScript, cont.

- Example 3

```
ScriptEngineManager manager =
   new ScriptEngineManager();
ScriptEngine engine =
manager.getEngineByName("js");
String script = Util.readFile("example1.js");

engine.eval(script);
Invocable inv = (Invocable) engine;

inv.invokeFunction("add",
   new Object[]{
      Integer.valueOf(12), Integer.valueOf(24)
   });
```

Resources

- http://quirksmode.com
- http://www.w3schools.com/js
- https://scripting.dev.java.net
- http://getfirebug.com
- http://developer.yahoo.com/yui
- http://www.dojotoolkit.org
- http://www.prototypejs.org/

JavaOne℠

Thank You

Jason Lee
jason@steeplesoft.com
jasondlee@sun.com
http://blogs.steeplesoft.com