CS 683 Emerging Technologies
Fall Semester, 2004
Doc 4 JavaScript
Contents

References

JavaScript: The Definitive Guide, 4th Ed., David Flanagan, O'Reilly, 2002

ECMAScript Language Specification v3, Dec 1999, http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf

JavaScript
Java and JavaScript

- Completely different languages
- Both use some C syntax

Versions Of JavaScript

JavaScript – Netscape's version

Jscript – Microsoft's version

ECMAScript – ECMA's standard for the language

http://www.ecma-international.org/publications/standards/Ecma-262.htm

Version	
JavaScript 1.0	Netscape 2
JavaScript 1.1	Netscape 3
JavaScript 1.2	Netscape 4, almost ECMA v1
JavaScript 1.3	Netscape 4.5, ECMA v1
JavaScript 1.4	Only on Netscape servers
JavaScript 1.5	Mozilla, Netscape 6, ECMA v3
Jscript 1.0	Close to JavaScript 1.0, IE 3
Jscript 2.0	Close to JavaScript 1.1, Late IE 3
Jscript 3.0	Close to JavaScript 1.2, ECMA v1, IE 4
JScript 4.0	Not in any Web browser
JScript 5.0	Partially EMCA v3, IE 5
JScript 5.5	ECMA v3, Close to JavaScript 1.5, IE 5.5
JScript 5.6	IE 6, for client side same as JScript 5.5
ECMA v1	First standard
ECMA v2	Adds some clarifications
ECMA v3	Adds switch, regular expressions, exception handling

Running JavaScript

In Web Browser

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
      "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
   <meta http-equiv="content-type" content="text/html;
        charset=iso-8859-1" />
   <title>Hello World Example</title>
</head>

<body>

<script type="text/javascript" language="javascript1.5">
   document.write('<h1>Hello World</h1>');
</script>

</body>
</html>
```

The above page when rendered in a browser that supports Javascript 1.5 and has javascript running will display Hello World as an h1 element

Html 4.0 depreciated the language attribute in favor of the type attribute. However the language attribute is perhaps better supported in browsers.

Running JavaScript without a Browser

Java interpreters exist outside of web browsers

Rhino

- Open source Javascript
- http://www.mozilla.org/rhino/
- Usually embedded in other applications

Sample Program

```
print("Hello World");
```

Web browsers do not support the print() method
Rhino does not support document.write()

Reserved Words

break	else	new	var
case	finally	return	void
catch	for	switch	while
continue	function	this	with
default	if	throw	delete
in	try	do	instanceof
typeof			

Future Reserved Words

abstract	enum	int	short
boolean	export	interface	static
byte	extends	long	super
char	final	native	synchronized
class	float	package	throws
const	goto	private	transient
debugger	implements	protected	volatile
double	import	public	

Basic Syntax

```
/* Comment */
var j = "This is a string"; // doulble or single
// var is optional
k = j + ' adding to a string'; // quotes indicate a string
print( k);

k = 2 * 3;    //The string is garbage collected

j = 12.34        //variables have no type
var hex = 0xff;

var trouble = 011; //Octal or base 10 ?
print( trouble);  //prints 9, may print 11

// Float literals
a = 3.14         // ; is optional
b = 6.03e12

// ; not optional between statements on same line
c = 12; d = 'cat " man'   //embeded " in a string
```

For more information about types see section 8 (page 24) ECMAScript Language Specification v3

Case Sensitive

JavaScript is case sensitive

JavaScript in IE (PC version) reportedly is not case sensitive

Undefind vesrse not defined

```
var x;
print( x);     //prints undefined

print( z);     //Exception raised – z has not been defined
```

If a variable as been defined and has not been assigned a value, its value is undefined. If you try to use a variable (as a parameter or on the right hand side of a = before it is defined an exception will be raised. Assigning a value to a variable will define the variable if it has not already been defined.

Strings

```
e = "first line\n second line";

print( e.length);    // 23
print( e.charAt(0) );     //f

print( 1 + 2 + 'cat');      //3cat
print('cat' + 1 + 2 );      //cat12

sub = e.substring(6,9);
print(sub);                         //lin

s = e.indexOf('s');
print(s);                           //3

substringIndex = e.indexOf('line');
print(substringIndex);                  //6

f = "First line here";
f.toUpperCase();                        //FIRST LINE HERE
f.toLowerCase();                        //first line here

f.split(' ');             //array of three strings 'First', 'line', 'here'
f.split("");              //'F','i','s','t'.',' ','l','i','n','e',' 'h', 'e', 'r', 'e'
f.split("", 5);
f.split("line");      //'First ', ' here'
```

For details and more string methods see page 98 of ECMAScript Language Specification v3

Non Standard String Methods

'cat'.bold() //cat

'cat'.anchor('sam') //cat

'cat'.link('index.html') //cat

'cat'.fontcolor('red') //cat

Special Numbers

Infintity	infinity
NaN	not a number
Number.MAX_VALUE	Largest number
Number.MIN_VALUE	Number closest to zero
Number.NaN	not a number
Number.POSITIVE_INFINITY	+inifinity
Number.NEGATIVE_INFINITY	-inifinity

Math Functions

Math.sin(4);

For more Math methods see section 15.8 (page 112) of ECMAScript Language Specification v3

Basic Control Structures

Same as C/C++/Java

```
if (1) print('1 = true')
if (true) print('true = 1')
a = 1; b = 3;
if ( a == b) print('ok')
if ( a === b) print('ok')
```

For details on control structures see section 12 (pages 61-69) of ECMAScript Language Specification v3

== verses ===

== are the two values the same

=== do the two items point to the same location

If

```
var a = 3;
if ( a == 1) {
   print( 'one');
}
else if (a == 2) {
   print( 'two');
}
else {
   print( 'large');
}
```

switch

```
var a;

switch (a) {
   case 1:
      print('one');
      break;
   case 2:
      print('one');
      break;
   case 2 + 1:
      print('three');
      break;
   default:
      print('large');
      break;
};
```

Nonnumeric Switches

```
var a;

switch (typeof a) {
   case 'number':
      print('number');
      break;
   case 'string':
      print('string');
      break;
   case 'boolean':
      print('boolean');
      break;
   case 'object':
      print('boolean');
      break;
   default:
      print('other');
      break;
};
```

This is a silly example as it can be shortened to print(typeof(a));

While, Do and for

```
var a = 2;

while ( a < 5 )
   {
   print( a++);
   }

do
   {
   print(a--);
   }
while (a > 0);

for (var count = 0; count < 5; count++)
   print( count);
```

Arrays
Creating Arrays

```
var a = new Array();
print(a.length);              //0
print(a[0]);                  //undefined

var b = new Array(1,2,3, 'cat');
print(b.length);              //4
print(b[0]);                  //1
print( b.toString() );        //1,2,3,cat

var c = new Array(10);
print(c.length);              //10
print(c[0]);                  //undefined

var d = [1, 2, true, 'dog'];
print(d.length);              //4
print(d[0]);                  //1

for (x in d)
   print(x);
```

For complete details on arrays see section 15.4 (page 88) of ECMAScript Language Specification v3

Accessing Array Elements

```
var a = new Array(5);
print(a.length);    //5

a[0] = 0;
a[10] = 10;
print(a.length);    //11

for (x in a)
   print(x);
```

Output

```
0
10
```

```
for (var x = 0; x < a.length; x++)
   print(a[x]);
```

Output

```
0
undefined
undefined
undefined
undefined
undefined
undefined
undefined
undefined
undefined
undefined
10
```

Accessing Array Elements

```
var a = new Array(5);
print(a.length);     //5

a[0] = 0;
a[10] = 10;

for (var x = 0; x < a.length; x++)
   if( a[x] != undefined) print(a[x]);
```

Output

```
0
10
```

```
print(a[20]);
```

Output

```
undefined
```

Some Array Methods
Join, reverse, sort

```
var a = [1, 2, 3];
var aString = a.join();
print(aString);                    //1,2,3

var bString = a.join("; ");
print(bString);                    //1; 2; 3

var reversed = a.reverse();
print(reversed);                   //3,2,1

var b = ['cat', 'and', 'bat'];
b.sort();
print(b);                          //and,bat,cat

var c = [33, 4, 222, 1111];
c.sort();
print(c);                          //1111,222,33,4

c.sort(function(a,b) { return a-b;});
print(c);                          //4,33,222,1111
```

Sort uses alphabetic order unless given a function

Concat

```
var a = [1, 2, 3];
b = a.concat(4, 5);
print(b);                       // 1,2,3,4,5

b = a.concat(4, 5, 6, 7);
print(b);                               // 1,2,3,4,5,6,7

b = a.concat([4,5], [6,7]);
print(b.length);                        // 7

b = a.concat(4, [5, [6, 7]]);
print(b.length);                        //6
print(b[5]);                            //6,7
```

Pop & push

```
var a = [ ];
a.push(1, 2);
print(a);           //1,2
a.pop();
print(a);           //1
a.push( 3);
print(a);           //1,3
a.push( [4,5]);
a.pop();            //returns [4,5]
```

slice

```
var a = [1, 2, 3, 4, 5];

a.slice(0, 2);     //returns 1,2
a.slice(3);      //returns 4,5

a.slice(1,-1);     //returns 2,3,4
a.slice(-4, -2);   //returns 2,3
```

slice(start) – from location start to end
slice(start, end) – from location start to location end

negative values for start and end indicate location from the end of the array

splice

Cuts elements from an array
Returns the elements cut
Modifies the array it acts on

```
var a = [1, 2, 3, 4, 5, 6, 7];
a.splice( 4);                    //first argument – were to start cut
//returns 5,6,7
//a = [1,2,3,4]

var a = [1, 2, 3, 4, 5, 6, 7];
a.splice(2, 4);                  //second argument – length of cut
//returns 3,4,5,6
// a=[1,2,7]

var a = [1, 2, 3, 4, 5, 6, 7];
a.splice(2, 0, 'a', 'b');
//returns [ ]
// a=[1,2,'a', 'b', 3, 4, 5, 6, 7]

var a = [1, 2, 3, 4, 5, 6, 7];
a.splice(2, 2, 'a', 'b');
//returns 3,4
// a=[1,2,'a','b',5,6,7]

var a = [1, 2, 3, 4, 5, 6, 7];
a.splice(2, 2, 'a', 'b', 'c');
//returns 3,4
// a=[1,2,'a','b','c',5,6,7]
```

Functions

```
function sum(n)
   {
   if (n <= 1)
      return n;
   return n + sum(n -1);
   }

function print(aString)
   {
   document.write(aString, "<br>");
   }

function hypotenuse(a, b)
   {
   function square(aNumber)
      {
      return aNumber * aNumber;
      }

   return Math.sqrt(square(a) + square(b));
   }

sum(3)    //6

hypotenuse(1, 1)          // 1.4142135623730951
```

Functions as Data

```
function square(aNumber)
   {
   return aNumber * aNumber;
   }

square(2)       //4

var fun = square
fun(2)          //4

function perform(aFunction, aValue)
   {
   return aFunction(aValue);
   }

perform(fun, 2)                 //4

perform(square, 2)                  //4

perform(Math.sin, 2)                //0.9092974268256817
```

Defining Functions

```
var x = function(n)
   {
   return n + 1;
   }

var y = function sum(n)
   {
   if (n <= 1) return n;
   return n + sum(n-1);
   }

var z = new Function('x', 'return x*x;');

x(2);     //3
y(3);     //6
z(2);     //4
```

Arguments

```
function sum()
   {
   var sum = 0;
   for (var k =0; k < arguments.length; k++)
      sum = sum + arguments[k];
   return sum;
   }

sum(1, 2, 3)      //6
sum(1,2 )         //3
```

callee

```
var u = function(n)
   {
   if (n <= 1) return n;
   return n + arguments.callee(n-1);
   }

u(3)      //6
```

Names & Functions

```
var foo = 3;

print(foo);              //3

function foo(n)
   {
   return n + 1;
   }

print(foo);              //function foo(n) { return n +1;}
```

Functions and Scope

```
var foo = 'global';

function scope()
   {
   var foo = 'local';
   print(foo);
   }

scope();          //local
print(foo);       //global

var foo = 'global';

function scope()
   {
   foo = 'local';
   print(foo);
   }

scope();          //local
print(foo);       //local
```

When you declare a variable in a function with a var it creates a local variable. If you use a variable in a function with declaring it, the variable is global

More Functions and Scope

```
function scope()
   {
   bar = 'local';
   print(bar);
   }

scope();
print(bar);
```