CS 696 Emerging Web and Mobile Technologies
Spring Semester, 2011
Doc 3 JavaScript
Jan 25, 2011

References

Javascript Tutorial, http://www.w3schools.com/js/default.asp

ECMA-262, Edition 5, http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf

CS 683 Lecture notes, http://www.eli.sdsu.edu/courses/fall04/cs683/notes/index.html

JavaScript

Netscape 1996

Scripting language for webpages

jScript

Microsoft's version

ECMAScript

ECMA European Standards body name

Prototype based scripting language

Dynamic

C like syntax

Not related to Java

WebServer in JavaScript

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(8124, "127.0.0.1");
console.log('Server running at http://127.0.0.1:8124/');
```

Uses Node.js http library

JS is now used on server side and is embedded into applications

Server side & Embed-able JS

Node.js

- Scalable network JS programs
- Server-side executable scripts

V8 JavaScript Engine

- Googles open source JS engine
- Standalone
- Embed in C++ programs

Rhino

- Standalone
- Embed in Java programs
- Included in Java 6

Running JavaScript

Desktop/Server side - Node.js

Download & install Node.js

hello.js

```
console.log('Hello World')
```

```
node hello.js
```


Runs on Linux, Mac, Unix,Windows/Cygwin but not on straight windows

In Web Browser

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="utf-8" />
	<title>Javascript.html</title>
</head>
<body>
<script type="text/javascript">
<!--
document.writeln( "Hello World",'<br/>');
//-->
</script>
</body>
</html>
```

Some Options

```
<!DOCTYPE html>
<html lang="en">
<head>
      <meta charset="utf-8" />
      <title>Javascript.html</title>
</head>
<body>
<script>
     var a = 2;
     debugger;
     while ( a < 5 ) {
          alert( a++);
     }
</script>
</body>
</html>
```


To use the javascript debugger:
Chrome: View - Developer - Developer Tools then Script tab
Safari: Turn on developer menu the Develop - Start Debugging Javascript
FireFox: Install Firebug

Using a file

JavaScript.html

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="utf-8" />
	<title>Javascript.html</title>
</head>
<body>


<script src="sample.js">
</script>

</body>
</html>
```

sample.js

```
document.writeln( "Hello World!",'<br/>');
```

Basic Syntax

Basic Syntax

```
/* Comment */
var j = "This is a string";
j = 'This is a string too';

// var is optional
k = j + ' adding to a string';
require('sys').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 ?
require('sys').print( trouble);  //prints 9, may print 11
```

More Basic Syntax

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

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

Case Sensitive

```
var k = 5;
var K = 10;
```

Undefined verses not Defined

```
var x;          //x is undefined

//z is not defined
```

Data Types & Built-in Objects

Undefined (undefined)
Null (null)
Boolean
    true, false
String
Number

Objects

Global
Object
Function
Array
String
Boolean
Number
Math
Date
RegExp
Error
JSON

String operations

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

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

1 + 2 + 'cat';          //3cat
'cat' + 1 + 2;          //cat12
```

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

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

e.indexOf('line');       //6
```

String Operations

```
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’,'r',’s’,’t’,‘ ’,’l’,’i’,’n’,’e’,’ ‘h’, ‘e’, ‘r’, ‘e’
f.split('', 5);                     //’F’,’i’,'r',’s’,’t’
f.split(“line”);                    //’First ‘, ‘ here’
```

String HTML Wrapper Methods

Not part of the standard

Support is Browser dependent

```
'cat'.bold()                //<b>cat</b>
'cat'.anchor('sam')         //<a name="sam">cat</a>
'cat'.link('index.html')    //<a href="index.html">cat</a>
'cat'.fontcolor('red')      //<font color="red">cat</font>
```

anchor()
big()
blink()
bold()
fixed()
fontcolor()
fontsize()
italics()
link()
small()
strike()
sub()
sup()

String - Number Conversion

```
1 + '2'              //12
1 + (+'2')           //3
'1' + 2              //12

parseFloat('2')          //2
parseFloat(' 2k')        //2
parseFloat(' m2k')       //Nan

parseInt('2')            //2
parseInt('2.34')             //2
```

Equality

== are the two values the same

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

Standard C Control Structures

```
var a = 3;

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

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

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

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

Switch

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

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

Arrays - Creating

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

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

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

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

Array - Accessing

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

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

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

for (x in a)
      if( a[x] != undefined) alert(a[x]);
```

Array Methods - Join, Reverse, Sort

```
var a = ['cat', 'and', 'bat'];;
var aString = a.join();                  //cat,and,bat

var bString = a.join("; ");              //cat; and; bat

var reversed = a.reverse();              //[ 'bat', 'and', 'cat' ]

a.sort();                                //[ 'and', 'bat', 'cat' ]
```

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

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

Array - concat

join two or more arrays

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

b = a.concat([4,5], [6,7], [8, 9]);    //[ 1, 2, 3, 4, 5, 6, 7, 8, 9 ]

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

Array - pop & push

```
var a = [ ];
a.push(1, 2);          //a = [1, 2]

a.pop();               // returns 2

a.push( 3);            // a = [1, 3]

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

Arrays - 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

Functions

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

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

```
sum(3)                  //6

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

```
function hypotenuse(a, b){
      function square(aNumber) {
            return aNumber * aNumber;
      }
      return Math.sqrt(square(a) + square(b));
}
```

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
```

Functions and Scope

```
var foo = 'global';

function scope()  {
      var foo = 'local';
      console.log(foo);
}

scope();                  //local
console.log(foo);         //global
```

```
var foo = 'global';

function scope() {
      foo = 'local';
      console.log(foo);
}

scope();                  //local
console.log(foo);         //global
```


console.log = Node.js output

More on Scope

```
function scope() {
        bar = 'local';
        console.log(bar);
        }

scope();
console.log(bar);
```

Context

functions remember the context in which they were defined

```
var a = 'global';
function test()  {
      console.log( a);
}

function perform(aFunction ) {
      var a = 'local';
      aFunction();
}

perform(test);          //output global
```

More Context

functions remember the context in which they were defined

```
var b = 'global';

function give( ) {
        var b = 'local';
        function inner()  {
                console.log( b );
        }
        return inner;
}

var test = give();
test();                         // prints local
```

Exceptions

```
try{
        console.log( 'Before');
        factorial(5);
        console.log( 'After');
}
catch (exception ) {
        console.log(exception);
}
finally {
        console.log( 'Done');
}
```

```
try{
        console.log( 'Before');
        factorial(5);
        console.log( 'After');
}
finally {
        console.log( 'Done');
}
```

```
try{
        console.log( 'Before');
        factorial(5);
        console.log( 'After');
}
catch (exception ) {
        console.log(exception);
}
```

Throwing an Exception

throw new Error(“a foobar occurred”);

throw 5;

throw ‘Cat’;

Objects

Creating Objects

```
var anObject = new Object();

var now = new Date();              //now

var semesterStart = new Date(2004, 8, 30);

var circle = { x: 0, y:0, radius: 3 }

console.log( circle.x);
console.log( circle.y);
console.log( circle.radius);
```

Constructors

```
function Circle(x, y, radius) {
        this.x = x;
        this.y = y;
        this.radius = radius;
}

var y = new Circle(1, 2, 3);
```


Note the lack of a class definition

Properties

```
var circle = { x: 0, y:0, radius: 3 };

circle.x = 12;              //set

var z = circle.y;           // read

circle.theta = 1.342;   // create new property

var w = {z: circle, name: ‘sam’ };

w.z.radius;                 // nested properties

var a = circle[‘x’];        //read x

circle[‘x’] = 55;           //set x
```

x, y, radius
Properties of the object circle

Enumerating Properties

```
var circle = { x: 0, y:0, radius: 3 }

for (var name in circle)
      console.log( name);
```

Output

x
y
radius

```
for (var name in circle)
      console.log( circle[name]);
```

Output

0
0
3

Methods - Per Object

```
function Circle(x, y, radius) {
        this.x = x;
        this.y = y;
        this.radius = radius;
}

function computeArea() {
        return Math.PI * this.radius * this.radius;
}
```

```
var circle = new Circle(0, 0, 3);

circle.area = computeArea;
circle.area(); //28.274

var x = new Circle(0, 0, 3);

x.area()        //raises an exception
```

Methods - In Constructor

```
function computeArea() {
        return Math.PI * this.radius * this.radius;
}

function Circle(x, y, radius) {
        this.x = x;
        this.y = y;
        this.radius = radius;
        this.area = computeArea;
}
```

```
var x = new Circle(0, 0, 3);
x.area()                    //28.274
```


Tuesday, January 25, 2011

Now every circle object will contain an area property. All of them will contain the same value. This is a waste of space

Prototype

```
function computeArea() {
      return Math.PI * this.radius * this.radius;
      }

function Circle(x, y, radius) {
      this.x = x;
      this.y = y;
      this.radius = radius;
}

Circle.prototype.area = computeArea;
Circle.prototype.pi = 3.1415;
```

```
var x = new Circle(0, 1, 3);

x.area();
x.pi;
```


Only one property for all Circle objects

Prototype

```
function computeArea() {
      return Math.PI * this.radius * this.radius;
      }

function Circle(x, y, radius) {
      this.x = x;
      this.y = y;
      this.radius = radius;
   Circle.prototype.area = computeArea;
   Circle.prototype.pi = 3.1415;
}
```

```
var x = new Circle(0, 1, 3);

x.area();
x.pi;
```


You can set prototype values in constructor

Prototype

Each object has a prototype

Constructors define new types of objects

An object's prototype starts as a clone of Object's prototype

Reading a Property

```
circle.x;
```

If the object has the property return its value

If not check the object's prototype for the property and return its value

If don't find the property it is not defined

Simulating Class Methods

```
function increase(n) {return n +1;};

function Circle(x, y, radius){
      this.x = x;
      this.y = y;
      this.radius = radius;
      }

Circle.PI = 3.1425;
Circle.increase = increase;
```

```
Circle.PI;
Circle.increase(3 );

var x = new Circle(1, 2, 3);

x.PI;                          //not defined
x.increase(3);          //Exception thrown
```

Inheritance

```
function Rectangle(centerX, centerY, height, width){
        this.x = centerX;
        this.y = centerY;
        this.height = height;
        this.width = width;
}

Rectangle.prototype.area = function()
                {return this.height * this.width; }
```

```
Square.prototype = new Rectangle(0,0,0,0);
Square.prototype.constructor = Square;

var x = new Square(0,0,2);
x.area();
```

```
function Square(centerX, centerY, height) {
        this.x = centerX;
        this.y = centerY;
        this.height = height;
        this.width = height;
}
```