CS 683 Emerging Technologies
Fall Semester, 2004
Doc 5 JavaScript part 2
Contents

References 2
Exceptions 5
Objects 6
Properties 8
Methods 10
Prototypes 12
Inheritance 15

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

Remembering the Context

```
var a = 'global';
function test()
   {
   print( a);
   }

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

perform(test);
```

Code prints global

More Context

```
var b = 'global';

function give( )
   {
   var b = 'local';
   function inner()
      {
      print( b );
      }
   return inner;
   }

var test = give();
test();
```

Code prints local

Exceptions

```
try
   {
   var n = prompt( “Enter positive integer” , “”);
   var result = factorial(n);
   alert(‘Answer is: ‘ + result);
   }
catch (exception )
   {
   alert(exception);
   }
finally
   {
   alert( ‘Done’);
   }
```

Throwing an Exception

```
throw new Error(“a foobar occurred”);

throw 5;

throw ‘Cat’;
```

Objects

Creation

```
var anObject = new Object();

var now = new Date();        //now

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

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

print( circle.x);
print( circle.y);
print( circle.radius);
```

Constructors

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

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

Properties

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

x, y, radius are properties of the object circle

```
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
```

Enumerating Properties

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

for (var name in circle)
   print( name);
```

Output

```
x
y
radius
```

```
for (var name in circle)
   print( circle[name]);
```

Output

```
0
0
3
```

Methods

```
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 Constructors

```
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
```

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

Prototypes

```
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;
```

Prototype Rules

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

Writing a Property

```
circle.x = 23;
```

If the object has the property set the value

If not create the property in the object and set it.

Simulating Class Methods & Properties

```
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;            //returns 3.1425

Circle.increase(3 );            //returns 4


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; }

Rectangle.prototype.isSquare = function()  {return false; }

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

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

Square.prototype.isSquare = function()   {return true; }

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

Object Properties & Methods

toString()
toLocaleString()
valueOf()