JAVASCRIPT Prototypes

Leslie Carr

COMP3001

JavaScript: Objects & Classes

…and functions and arrays

They’re all the same really.

JavaScript Objects

- A JavaScript object has properties associated with it.

  objectName.*propertyName*

- Define a property by assigning it a value

```
myCar.make = "Ford";
myCar.model = "Mustang";
myCar.year = 1969;
```

JavaScript Arrays, err, Objects

- An array is an ordered set of values associated with a single variable name.
- Properties and arrays are different interfaces to the same data structure.

```
myCar["make"] = "Ford"
myCar["model"] = "Mustang”
myCar["year"] = 1967
```

- NB array subscripts can contain illegal object property characters e.g. space

Creating an Array

- Either use an array constructor

```
arrayObjectName = new Array(element0, element1, …)
arrayObjectName = new Array(arrayLength)
```

- Or use an array literal

```
coffees = ["French Roast", "Columbian", "Kona"]
```

- Predefined array field

`array.length` current maximum size of array.

Creating a New Function

- Use an function declaration:

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

- Or a function expression:

```
square = function (number) {
   return number * number;}
```

- Or a function constructor:

```
multiply = new Function("x", "y", "return x * y")
```

- e.g.

```
map(function(x) {return x * x * x}, [0, 1, 2, 5, 10]);
```

Creating a New Object

- Use the builtin object types

- var today = new Date()
- var xmas = new Date(2007,11,25)
- var myObj = new Object()

Creating a New Object

- Use an object initializer:

```
objectName = {property1:value1, property2:value2,
  property3:value3, …}
```

- Create `myHonda` object with 3 properties.

```
myHonda = {color:"red", wheels:4,
  engine:{cylinders:4, size:2.2} }
```

- Note that the third property is an object in its own right.

Creating a New Object

- Alternatively,
  - Define the object type by writing a constructor function that specifies its name, properties, and methods.
  - Create an instance of the object with `new`.

```
function car(make, model, year){
 this.make = make;
 this.model = model;
 this.year = year;
 }
mycar = new car("Eagle", "Talon TSi", 1993);
```

- Create methods by assigning function expressions as property values.

JavaScript Has No Classes!

- In Java, all objects are made by instantiating class definitions
- In JavaScript, objects are made manually, by adding property/value pairs to an empty object
- Constructors help you do this automatically
- Prototypes (see next slide) let you inherit missing fields (class variables, methods) from other objects.
  - “class” constructor functions
  - instances of “superclass” objects

Creating an Object Prototype

- You can add a property to a *previously defined* object type by using the `prototype` property.
- This defines a property that is shared by all objects of the specified type.
- The protoype is a property of the object constructor function

```
car.prototype.color=null;  //”class” variable
car1.color="black";        //”object” value
```

Inheritance via Prototypes

- JavaScript objects inherit properties from a *prototype object*.
- If a property is not found in an object then its prototype property is checked to see if it does have that property.
- If the prototype object does *not* have the property then *its* prototype is checked.
- The prototype for an object is set by the *prototype* property of the constructor function that was used to create and initialize the object.

Inheritance Example

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

Circle.prototype.pi = 3.14159

circumference() {return 2 * this.pi * this.r }
Circle.prototype.circumference = circumference

Circle.prototype.area = function () { return this.pi * this.r * this.r }
```

Example use
```
var c = new Circle(0.0,0.0,10.0);
    var a = c.area(); var p = c.circumference();
```

Subclassing Example

- To make a ‘subclass’
  - set the prototype property of the constructor function to be *an instance of* the ‘superclass’
  - Don’t foprget there are no classes so there are no real superclasses or subclasses!

```
Employee(){ this.name = ””;
  this.dept = "general" }
function Manager() { this.reports = [] }
Manager.prototype = new Employee();
```