software studio

this & new : nasty effects

Daniel Jackson

confused javascript

wants to be a prototyping language

› no classes, prototype chain

wants to be a standard OO language

› instead of cloning operator, has pseudo constructor

consequence

› some strange rules

› easy to mess up

this

this is dynamically scoped

› in evaluating e.m(), this is bound to value of e inside m

› but reverts to global environment in calls that m makes

```
var counter = {
  val: 0,
  inc: function() {this.val += 1; return this.val;}
}
counter.inc(); // 1
counter.inc(); // 2
```

where *this* fails

a point ADT:

```
var Point = function (x, y) {
    this.x = function () {return x;}
    this.y = function () {return y;}
    mag = function () {return Math.sqrt(x*x+y*y);}
    this.unit = function () {return new Point (x/mag(), y/mag());}
}
```

playing with it:

```
> p = new Point(3,4)
Point {x: function, y: function, unit: function}
> p.unit().y()
0.8
```

where *this* fails

with defaulting:

```
var Point = function (x, y) {
  this.x = function () {return x ? x : 0;}
  this.y = function () {return y ? y : 0;}
  mag = function () {return Math.sqrt(this.x() * this.x() + this.y() * this.y());}
  this.unit = function () {return new Point (this.x()/mag(), this.y()/mag());}
}
```

playing with it:

```
> p = new Point(2)
Point {x: function, y: function, unit: function}
> p.unit().y()
TypeError: Object [object global] has no method 'x'
```

fixed:

```
var Point = function (x, y) {
  var that = this;
  this.x = function () {return x ? x : 0;}
  this.y = function () {return y ? y : 0;}
  mag = function () {return Math.sqrt(that.x() * that.x() + that.y() * that.y());}
  this.unit = function () {that = this;
                           return new Point (this.x()/mag(), this.y()/mag());}
}
```

another *this* failure

abstract out logging:

```
> f = function (reporter) {
    for (var i = 0; i < 4; i++)
        reporter("Step " + i);
    }
> f(console.log);
TypeError: Illegal invocation
```

a workaround:

```
> r = function (s) {console.log(s);}
> f(r)
Step 0
Step 1
Step 2
Step 3
undefined
```

forgetting to use "new"

```
var Point = function (x, y) {
    this.x = function () {return x;}
    this.y = function () {return y;}
}
```

```
> x = 3
3
> var p = Point(1,2)
undefined
> p.x()
TypeError
> x
function () {return x;}
> p
undefined
```

a remedy:

```
var Point = function (x, y) {
    if (!(this instanceof Point)) return new Point(x, y);
    this.x = function () {return x;}
    this.y = function () {return y;}
}
```

so this or that?

how to make an ADT

- with *this* & *new*
- with closures alone

	this & new	*closures*
instanceof	yes	no
extend prototype	yes, but can't see rep	not so easily
avoid nasties	no	yes

directly selecting prototype

choose a prototype for each new object?

```
Object.create = function (o) {
    var F = function () {};
    F.prototype = o;
    return new F();
}
```

```
> color = {bits: 24}
Object
> red = Object.create(color)
Object.create.F
> red.r = 255; red.g = 0; red.b = 0;
0
> red.bits
24
```

Object.create: in ECMA 5, and implemented in most browsers

6.170 Software Studio
Spring 2013