objects, literals & constructors

Daniel Jackson
easiest way to make an object

```javascript
yellow = {red: 255, green: 255, blue: 0}
Object

toCSS = function (c) {
    return "rgb("
    + c.red + "," + c.green + "," + c.blue
    + ")"
}
```

function...

```javascript
document.body.style.backgroundColor = toCSS(yellow)
"rgb(255,255,0)"
```
getting & setting properties

> first setting creates slot

```javascript
> cyan = {};
Object
> cyan.red
undefined
> cyan.red = 0; cyan.green = 255; cyan.blue = 255
255
> cyan
Object
  1.blue: 255
  2.green: 255
  3.red: 0
  4.__proto__: Object
```
constructors, with literals

```javascript
var Color = function (r, g, b) {
    return {red: r, green: g, blue: b};
};
yellow = Color(255, 255, 0);
document.body.style.backgroundColor = toCSS(yellow)
```

> just a regular function
constructors, with this

```javascript
var Color = function (r, g, b) {
  this.red = r; this.green = g; this.blue = b;
}
red = new Color(255, 0, 0);
document.body.style.backgroundColor = toCSS(red)
```

also just a regular function, but when called with new
› allocates fresh object
› binds to variable this
› returns this at end of call

what happens if you forget ‘new’?
› binding of this unchanged
› default binding is to top level environment!
an object model of objects

- slot value can be anything, including function
methods

```
var Color = function (r, g, b) {
    this.red = r; this.green = g; this.blue = b;
    this.toCSS = function () {
        return "rgb(\" + this.red + ",\" + this.green
            + ",\" + this.blue + ")\""
    }
}
red = new Color(255, 0, 0);
document.body.style.backgroundColor = red.toCSS()
```

› just put function in object slot!
› how is this bound in method call?
   in evaluating e.m(), this is bound to value of e inside m