Javascript & Objects

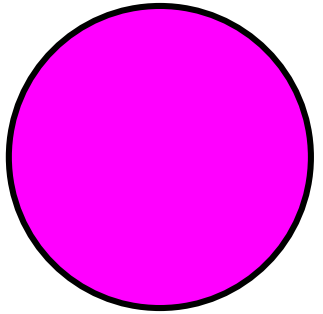
CSC 240 Supplemental Video

Nicholas R. Howe

Javascript Objects

In larger programs, it is often useful to group related data and actions.

- Example: object that will be displayed
  - Details about appearance
  - Details about location & pose
  - Functions for drawing, updating, etc.



- Javascript objects provide for a named set of values
  - Those holding data are called **properties**
  - Those holding actions are called **methods**
- Access via dot notation:

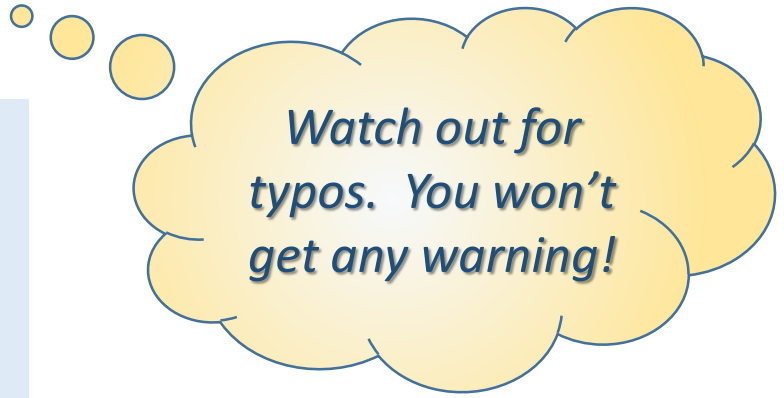
```
console.log(myCircle.area());
```

```
let myCircle = {
  radius: 32,
  color: "magenta",
  area: function() {
    return Math.PI*this.radius**2
  }
}
```

Using Javascript Objects

You can add and remove properties and methods at any time.

```
myCircle.edgeColor = "black";
delete myCircle.color;
myCircle.fillColor = "magenta";
myCircle.x = 100;
myCircle.y = 100;
```



Attempting to access a property that doesn't exist returns **undefined**.

Methods

Methods can access other properties and methods via **`this`**:

```
myCircle.draw = function(graphics) {
  graphics.beginPath();
  graphics.arc(this.x,this.y,this.radius,0,2*Math.PI);
  graphics.strokeStyle = this.edgeColor;
  graphics.stroke();
  graphics.fillStyle = this.fillColor;
  graphics.fill();
}
```

Constructors

A constructor simplifies the creation of many of similar objects

Without a constructor:

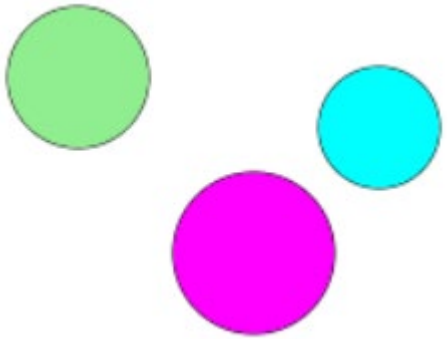
```
let myCircle = {
  radius: 32,
  color: "magenta",
  area: function() {
    return Math.PI*this.radius**2
  }
}
```

With a constructor:

```
// creates a new Circle object
function Circle(x,y,radius) {
  this.x = x;
  this.y = y;
  this.radius = radius;
  this.color = "cyan";
  this.area = function() {
    return Math.PI*radius**2;
  }
}

myCircle2 = new Circle(150,50,24);
myCircle3 = new Circle(30,30,28);
```

A Complete Script



```
// defines a new Circle object
function Circle(x,y,radius,color) {
  this.x = x;
  this.y = y;
  this.radius = radius;
  this.color = color;
  this.area = function() {
    return Math.PI*radius**2;
  }
  this.draw = function(graphics) {
    graphics.beginPath();
    graphics.arc(this.x,this.y,this.radius,0,2*Math.PI);
    graphics.strokeStyle = "black";
    graphics.stroke();
    graphics.fillStyle = this.color;
    graphics.fill();
  }
}
```

```
// set up array of Circle objects
var circles = [];
circles.push(new Circle(100,100,32,"magenta"));
circles.push(new Circle(150,50,24,"cyan"));
circles.push(new Circle(30,30,28,"lightgreen"));

// use the definitions above to draw
function draw(graphics) {
  for (c in circles) {
      circles[c].draw(graphics);
  }
}

function init() {
  canvas = document.getElementById("theCanvas");
  graphics = canvas.getContext("2d");
  draw(graphics);
}
```

Summary

- Objects can be created to bundle related data and actions
  - Access a particular property or method using dot notation
- The specific collection of properties and methods for an object can be modified at will
  - References to nonexistent ones will return undefined
- Methods can access other properties and methods via **`this`**
- A constructor simplifies the creation of many of similar objects