

INFM 603: Information Technology and Organizational Context

# **Session 5: JavaScript – Functions and Objects**



Jimmy Lin  
The iSchool  
University of Maryland

Thursday, October 4, 2012



It'll all make sense today...



**Programming... is a lot like cooking!**

# Functions

- Reusable code for performing a computation
- A function...
  - Takes in one or more parameters
  - Executes some code
  - Optionally returns a value

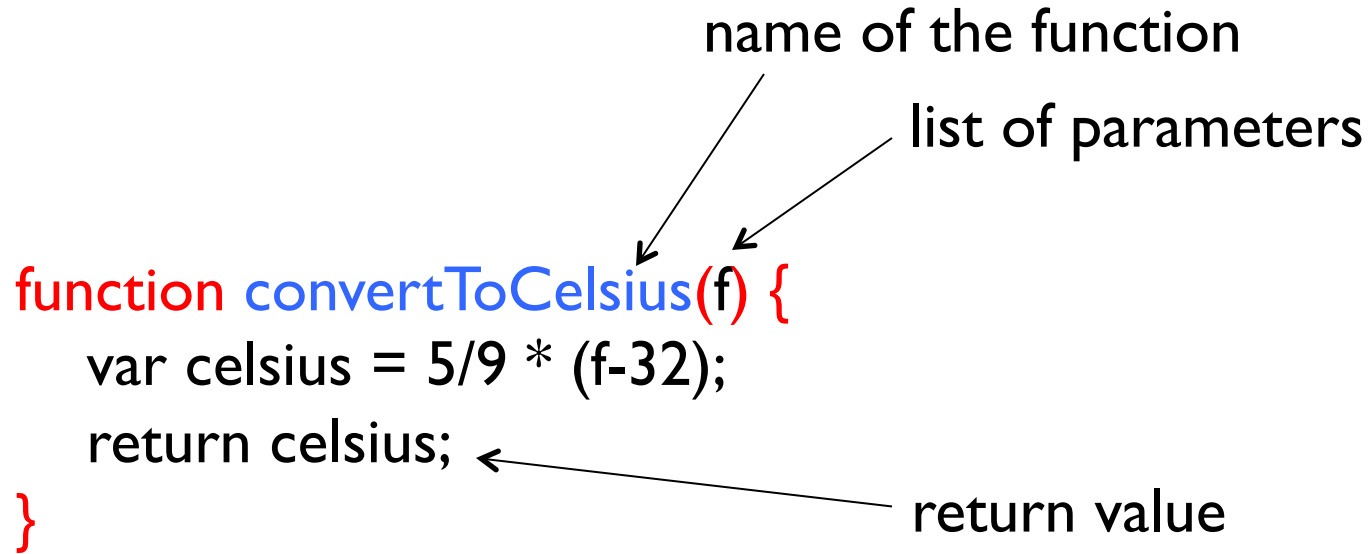
# Anatomy of Functions

name of the function

list of parameters

```
function convertToCelsius(f) {  
    var celsius = 5/9 * (f-32);  
    return celsius;  
}
```

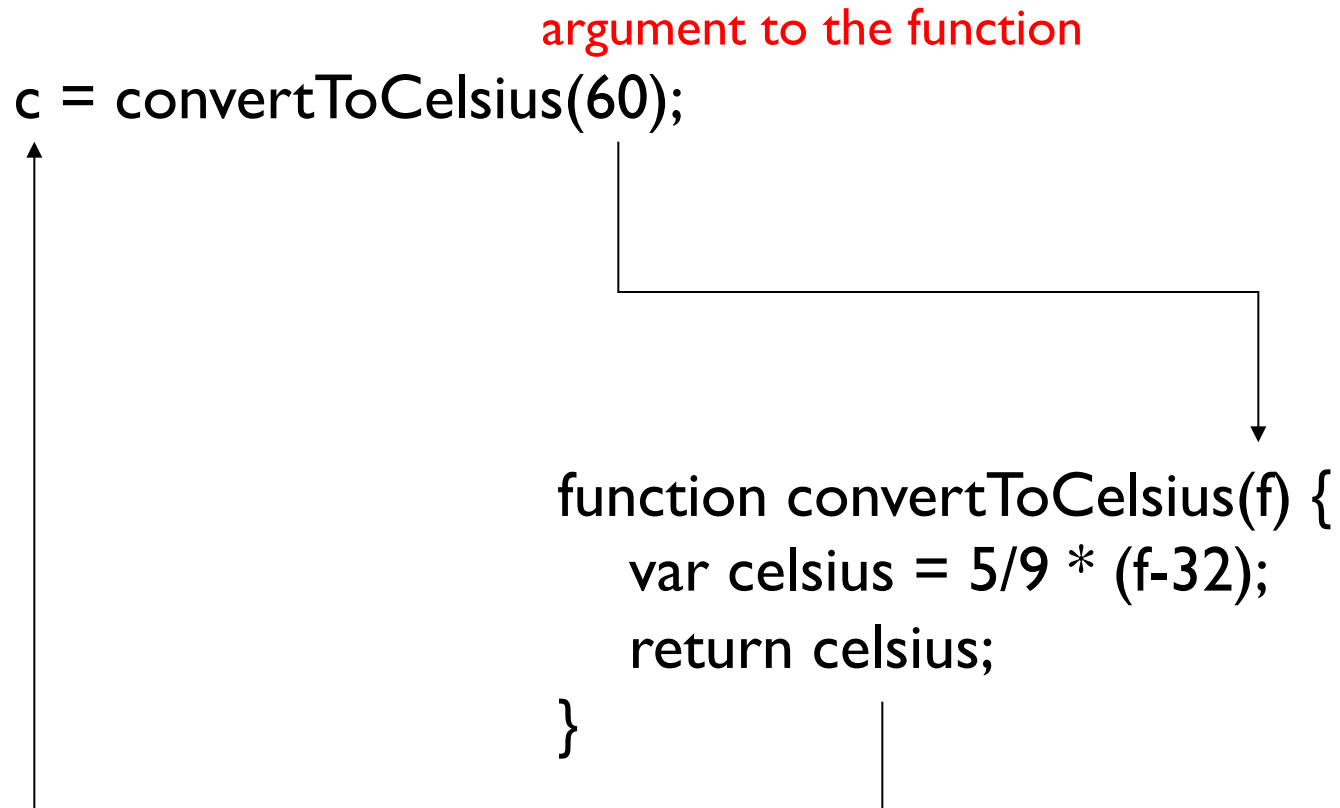
return value



```
function weirdAddition(a, b) {  
    var result = a + b - 0.5;  
    return result;  
}
```

# Using Functions

- Calling functions invokes the set of instructions it represents
  - Arguments to the function are specified between the parens
  - Multiple arguments are separated by commas



# More Examples

```
var r = weirdAddition(2, 4);
```

```
var a = 2;
```

```
var b = 3;
```

```
var s = weirdAddition(a, b);
```

```
function weirdAddition(a, b) {  
    var result = a + b - 0.5;  
    return result;  
}
```

# You've already been doing it!

- Built in functions:
  - `prompt("enter some text", "default");`
  - `alert("message here");`
- Message handlers!



Cooking analogy?

# Objects

- It's just a collection of properties!



```
var fido = {  
  name: "Fido",  
  weight: 40,  
  breed: "Mixed",  
  loves: ["walks", "fetching balls"]  
};
```

# Objects and Properties

- Access object properties using the “dot” notation

```
var w = fido.weight;  
fido.breed = "Yellow Lab";
```

- Where have we seen this before?



Wine demo (Part I)

# Objects: Methods

- It's just a collection of properties!
- Objects can have functions also! (called methods)



```
var fido = {  
  name: "Fido",  
  weight: 40,  
  breed: "Mixed",  
  loves: ["walks", "fetching balls"],  
  bark: function() {  
    alert("Woof woof!");  
  }  
};
```

# Calling a Method

- Invoke an object's method using the dot notation:

```
fido.bark();
```

- It's just like a function call!
- Where have you seen this before?
- What's "this"?

# What's the point?

- Claim: every method can be rewritten as a ordinary function, and vice versa?
- Why have methods? What's the advantage of functions directly to objects?



Wine demo (Part 2)

# Constructor!

- So far, building objects is a tedious process... that's where constructors come in:

```
function Dog(name, breed, weight) {  
  this.name = name;  
  this.breed = breed;  
  this.weight = weight;  
  this.bark = function() {  
    if (this.weight > 25) {  
      alert(this.name + " says Woof!");  
    } else {  
      alert(this.name + " says Yip!");  
    }  
  };  
}
```

# Using Constructors

- Invoke constructors using “new”:

```
var fido = new Dog("Fido", "Mixed", 38);  
var tiny = new Dog("Tiny", "Chawalla", 8);  
var clifford = new Dog("Clifford", "Bloodhound", 65);
```

```
fido.bark();  
tiny.bark();  
clifford.bark();
```



Wine demo (Part 3)





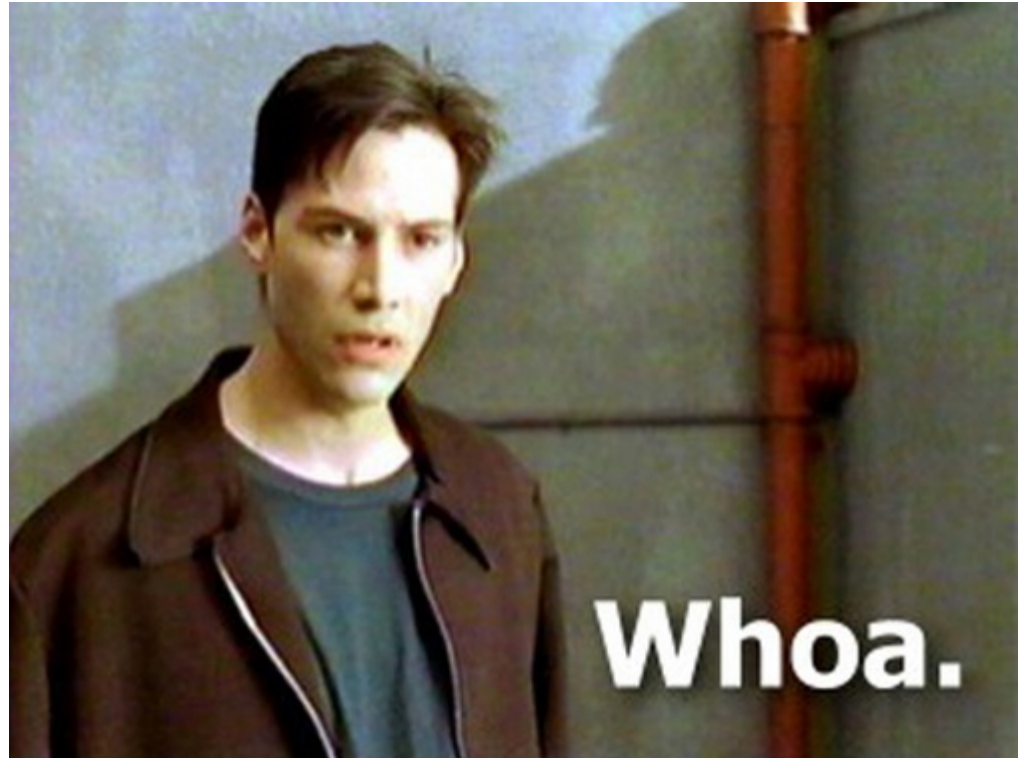
It'll all make sense today...



Make sense now?



Okay, you can relax now...



(a bit) more of this...

What happens if a function calls itself?



How do we find an element  
in a sorted list of elements?

How do we quantify the  
speed of various algorithms?

# Sequential Search





# Binary Search

Source: Wikipedia



# Algorithmic Complexity

- Linear vs. logarithmic algorithms
- Matters as data sizes increase!