

# Functions Part II

UC Berkeley Graduate School of Journalism

# Functions Review

```
function someName(argument)
{

    //code goes here

}
```

# Functions Review

```
function someName(argument)
{

    //code goes here

}

someName(data);
```

# What we know about Functions

so far....

- Functions don't run until you call them.
- JavaScript goes through your whole program before it runs, and captures the function declarations, so it knows about them **before** your code runs.
- This means you can call a function before you've even declared it!

# Variable Scope

```
function someName(){  
  
}
```

# Variable Scope

```
function someName(){  
    var myName = "Jeremy";  
}
```

# Variable Scope

```
function someName(){  
    var myName = "Jeremy";  
}  
  
someName();
```

# Variable Scope

```
function someName(){  
    var myName = "Jeremy";  
}
```

```
someName();  
$( '#response' ).html(myName);
```



# Variable Scope

```
function someName(){  
    var myName = "Jeremy";  
}
```

```
someName();  
$( '#response' ).html(myName);
```

**OUTPUT:** *Undefined*

# Variable Scope

```
function someName(){  
    var myName = "Jeremy";  
}
```

```
someName();  
$( '#response' ).html(myName);
```

# Variable Scope

```
var myName = "Jeremy";  
function someName()  
  
}  
  
someName();  
$( '#response' ).html(myName);
```

# Variable Scope

```
var myName = "Jeremy";  
function someName(){  
  
}  
  
someName();  
$( '#response' ).html(myName);
```

**OUTPUT: Jeremy**

What happens in a  
function, stays in a



is declared  
What ~~happens~~ in a  
function, stays in a



# Variable Scope

```
function someName(){  
    var myName = "Jeremy";  
}
```

```
someName();  
$( '#response' ).html(myName);
```

# Variable Scope

```
var myName = "";  
  
function someName(){  
    myName = "Jeremy";  
}  
  
someName();  
$( '#response' ).html(myName);
```



# Variable Scope

```
var myName = "";  
  
function someName(){  
    myName = "Jeremy";  
}  
  
someName();  
$( '#response' ).html(myName);
```

**OUTPUT: Jeremy**

# Pop Quiz

```
function someName(){  
    var i = 50;  
}  
$( '#response' ).html(i);
```

What is printed to the #response box?

```
function someName(){  
    var i = 50;  
}  
$( '#response' ).html(i);
```

What is printed to the #response box?

**Nothing**

```
var i;  
function someName(){  
    i = 50;  
}  
$( '#response' ).html(i);
```

What is printed to the #response box?

```
var i;  
function someName(){  
    i = 50;  
}  
$( '#response' ).html(i);
```

What is printed to the #response box?

**50**

```
var i, j, k;  
function someName(){  
    i = 50;  
    j = 50;  
    k = i + j;  
}  
$( '#response' ).html(k);
```

What is printed to the #response box?

```
var i, j, k;  
function someName(){  
    i = 50;  
    j = 50;  
    k = i + j;  
}  
$( '#response' ).html(k);
```

What is printed to the #response box?

**100**



```
var j;  
function someName(){  
    var i = 50;  
    j = i;  
}  
$( '#response' ).html(j);
```

What is printed to the #response box?

```
var j;  
function someName(){  
    var i = 50;  
    j = i;  
}  
$( '#response' ).html(j);
```

What is printed to the #response box?

**50**

```
var score = 0;
function someName(){
    var highscore = 100;
    if(score < highscore){
        score += 1;
    }
}
$('#response').html(score);
```

What is printed to the #response box?

```
var score = 0;
function someName(){
    var highscore = 100;
    if(score < highscore){
        score += 1;
    }
}
$('#response').html(score);
```

What is printed to the #response box?

**1**

```
function someName(){  
    var total = 100;  
    return total;  
}
```

```
$( '#response' ).html(someName());
```

What is printed to the #response box?

```
function someName(){  
    var total = 100;  
    return total;  
}
```

```
$( '#response' ).html(someName());
```

What is printed to the #response box?

**100**

# Assigning a function to a variable

```
var printJeremyStatus = function(){  
    return "Jeremy is pretty cool guy";  
}
```

# Assigning a function to a variable

```
var printJeremyStatus = function(){  
    return "Jeremy is pretty cool guy";  
}
```

**Anonymous function**



# Differences

```
function someName(){  
  
}
```

# Differences

```
function() {
```

```
}
```

# List of datatypes

34892

# List of datatypes

34892	Number
-------	--------

# List of datatypes

34892	Number
"Hey, this is neat"	

# List of datatypes

34892	Number
"Hey, this is neat"	String

# List of datatypes

34892	Number
"Hey, this is neat"	String
false	

# List of datatypes

34892	Number
"Hey, this is neat"	String
false	Boolean



# List of datatypes

34892	Number
"Hey, this is neat"	String
false	Boolean
[43, 24, "hi", true]	

# List of datatypes

34892	Number
"Hey, this is neat"	String
false	Boolean
[43, 24, "hi", true]	Array

# List of datatypes

34892	Number
"Hey, this is neat"	String
false	Boolean
[43, 24, "hi", true]	Array
{"name": "Joe", age: 3}	

# List of datatypes

34892	Number
"Hey, this is neat"	String
false	Boolean
[43, 24, "hi", true]	Array
{"name": "Joe", age: 3}	Object Literal

# List of datatypes

34892	Number
"Hey, this is neat"	String
false	Boolean
[43, 24, "hi", true]	Array
{"name": "Joe", age: 3}	Object Literal
function(){ return 5; }	

# List of datatypes

34892	Number
"Hey, this is neat"	String
false	Boolean
[43, 24, "hi", true]	Array
{"name": "Joe", age: 3}	Object Literal
function(){ return 5; }	Function

# Anonymous Functions

```
$( '#response' ).html( )
```

# Anonymous Functions

```
$( '#response' ).html(
```

```
)
```



# Anonymous Functions

```
$( '#response' ).html(  
    function(){  
        var total = 0;  
        for(i=0; i < 1000; i++){  
            total += i;  
        }  
        return total;  
    }  
)
```

# Anonymous Functions

```
$( '#response' ).html(  
    function(){  
        var total = 0;  
        for(i=0; i < 1000; i++){  
            total += i;  
        }  
        return total;  
    }  
)
```

**499500**

# Functions as properties

```
var car = {
```

```
}
```

# Functions as properties

```
var car = {  
    "type" : "sedan",  
  
}
```

# Functions as properties

```
var car = {  
    "type"    : "sedan",  
    "make"    : "Toyota",  
  
}
```

# Functions as properties

```
var car = {  
  "type"    : "sedan",  
  "make"    : "Toyota",  
  "year"    : 2013,  
  
}
```

# Functions as properties

```
var car = {  
  "type"    : "sedan",  
  "make"    : "Toyota",  
  "year"    : 2013,  
  "start"   : function(){  
  
  }  
}
```

# Functions as properties

```
var car = {  
  "type" : "sedan",  
  "make" : "Toyota",  
  "year" : 2013,  
  "start" : function(){  
  
  }  
}
```

```
car.type  
car.make  
car.year  
car.start()
```



# Object Oriented Programming

# Object Oriented Programming

```
car.color  
car.make  
car.size  
car.hasFuel  
car.type
```

# Object Oriented Programming

```
car.color  
car.make  
car.size  
car.hasFuel  
car.type
```

adjectives

# Object Oriented Programming

## **properties**

```
car.color  
car.make  
car.size  
car.hasFuel  
car.type
```

adjectives

# Object Oriented Programming

## **properties**

```
car.color  
car.make  
car.size  
car.hasFuel  
car.type
```

adjectives

```
car.start()  
car.go()  
car.stop()  
car.playMusic()  
car.breakDown()
```

# Object Oriented Programming

## **properties**

```
car.color  
car.make  
car.size  
car.hasFuel  
car.type
```

adjectives

```
car.start()  
car.go()  
car.stop()  
car.playMusic()  
car.breakDown()
```

verbs

# Object Oriented Programming

## properties

```
car.color  
car.make  
car.size  
car.hasFuel  
car.type
```

adjectives

## methods

```
car.start()  
car.go()  
car.stop()  
car.playMusic()  
car.breakDown()
```

verbs

# Object Oriented Programming

## properties

```
car.color  
car.make  
car.size  
car.hasFuel  
car.type
```

adjectives

## methods

```
car.start()  
car.go()  
car.stop()  
car.playMusic()  
car.breakDown()
```

verbs

## events

```
car.addEventListener('move')  
car.addEventListener('breakDown')  
car.addEventListener('honk')
```



# Object Oriented Programming

# Object Oriented Programming

```
video.src = "greatfilm.mp4";  
video.volume = 75;  
video.autoPlay = false;
```

# Object Oriented Programming

```
video.src = "greatfilm.mp4";  
video.volume = 75;  
video.autoPlay = false;
```

```
video.load();  
video.play();  
video.pause();
```

# Object Oriented Programming

```
video.src = "greatfilm.mp4";  
video.volume = 75;  
video.autoPlay = false;
```

```
video.load();  
video.play();  
video.pause();
```

```
video.on( 'play' );  
video.on( 'ended' );  
video.on( 'canplaythrough' );
```

# Object Oriented Programming

## **properties**

```
video.src = "greatfilm.mp4";  
video.volume = 75;  
video.autoPlay = false;
```

```
video.load();  
video.play();  
video.pause();
```

```
video.on( 'play' );  
video.on( 'ended' );  
video.on( 'canplaythrough' );
```

# Object Oriented Programming

## **properties**

```
video.src = "greatfilm.mp4";  
video.volume = 75;  
video.autoPlay = false;
```

## **methods**

```
video.load();  
video.play();  
video.pause();
```

```
video.on( 'play' );  
video.on( 'ended' );  
video.on( 'canplaythrough' );
```

# Object Oriented Programming

## **properties**

```
video.src = "greatfilm.mp4";  
video.volume = 75;  
video.autoPlay = false;
```

## **methods**

```
video.load();  
video.play();  
video.pause();
```

## **events**

```
video.on( 'play' );  
video.on( 'ended' );  
video.on( 'canplaythrough' );
```

# Functions as properties

```
var car = {  
  "type"    : "sedan",  
  "make"    : "Toyota",  
  "year"    : 2013,  
  "start"   : function(){  
  
  }  
}
```



What is "this"?

**THIS**

# What is "this"?

```
var car = {  
  "type" : "sedan",  
  "make" : "Toyota",  
  "year" : 2013,  
  "start" : function(){  
    $( '#reponse' ).html(this.make)  
  }  
}
```

**"this" always refers to the thing that called the function**

# What is "this"?

```
var car = {  
  "type" : "sedan",  
  "make" : "Toyota",  
  "year" : 2013,  
  "start" : function(){  
    $('#reponse').html(this.make)  
  }  
}
```

"this" always refers to the thing that called the function



# What is "this"?

```
var car = {  
  "type" : "sedan",  
  "make" : "Toyota",  
  "year" : 2013,  
  "start" : function(){  
    $( '#reponse' ).html(this.make)  
  }  
}
```

"this" always refers to the thing that called the function

# Pop Quiz

```
var person = {  
    name      : "Joe",  
    greeting : function(){  
        $('#resp').html("Hello Mr. " + this.name);  
    }  
}  
  
person.greeting();
```

```
var person = {  
  name      : "Joe",  
  greeting : function(){  
    $('#resp').html("Hello Mr. " + this.name);  
  }  
}  
  
person.greeting();
```

**Hello Mr. Joe**

```
var person = {  
    name      : "Joe",  
    greeting : function(){  
        $('#resp').html("Hello Mr. " + this.name);  
    }  
}
```

```
person.name = "Thompson";  
person.greeting();
```



```
var person = {  
  name      : "Joe",  
  greeting : function(){  
    $('#resp').html("Hello Mr. " + this.name);  
  }  
}
```

```
person.name = "Thompson";  
person.greeting();
```

**Hello Mr. Thompson**

```
var car = {  
  make : "Chevy",  
  run : function(){  
    $( '#resp' ).html("Vrrooom " + this.make );  
  }  
}  
  
car.run();
```

```
var car = {  
  make : "Chevy",  
  run : function(){  
    $( '#resp' ).html("Vrrooom " + this.make );  
  }  
}  
  
car.run();
```

**Vrrooom Chevy**

```
var animal = {  
    makeSound : function(){  
        $( '#resp' ).html(this.call);  
    }  
}
```

```
animal.makeSound();
```

```
var animal = {  
    makeSound : function(){  
        $( '#resp' ).html(this.call);  
    }  
}
```

```
animal.makeSound();
```

*undefined*

```
var animal = {  
    makeSound : function(){  
        $( '#resp' ).html(this.call);  
    }  
}
```

```
animal.call = "Bark!"  
animal.makeSound();
```

```
var animal = {  
    makeSound : function(){  
        $( '#resp' ).html(this.call);  
    }  
}
```

```
animal.call = "Bark!"  
animal.makeSound();
```

**Bark!**

# Three things we learned:

1. Variables defined within a function only exist within that function.
2. Functions can be anonymous, and thus viewed as a datatype, just like strings/booleans, etc.... and can be assigned to variables if desired.
3. The word "this" refers to the parent scope in which the function is called.



# Type of Objects you've been using

```
jQuery.getJSON();
```

```
document.getElementById();
```

```
d3.select('div')
```

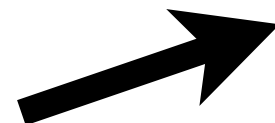
# Type of Objects you've been using



`jQuery`.getJSON();



`document`.getElementById();



`d3`.select('div')

# document object

`document.images`

`document.links`

`document.title`

`document.URL`

`document.cookie`

`document.getElementById()`

`document.write()`

`document.createElement()`

`click`

`onmousemove`

`onkeypress`

# document object

## **properties**

`document.images`

`document.links`

`document.title`

`document.URL`

`document.cookie`

`document.getElementById()`

`document.write()`

`document.createElement()`

`click`

`onmousemove`

`onkeypress`

# document object

## properties

`document.images`

`document.links`

`document.title`

`document.URL`

`document.cookie`

## methods

`document.getElementById()`

`document.write()`

`document.createElement()`

`click`

`onmousemove`

`onkeypress`

# document object

## properties

`document.images`

`document.links`

`document.title`

`document.URL`

`document.cookie`

## methods

`document.getElementById()`

`document.write()`

`document.createElement()`

## events

`click`

`onmousemove`

`onkeypress`