# Javascript Notes

May 23, 2013

## Contents

## 1 Scope

The global namespace is available as a global object. The *this* keyword intitally references this object.

Q: Generally global variables are bad, so why would you need to use the global scope?
A: To interact between seperate program components. To detect features of the host environment.

Q: How can you accidently create a global variable within a function?
A: By declaring without the *var* keyword

Q: Scope chain vs. Prototype chain?
A:

## 2 Functions

### 2.1 General

A function is a function. A method is a function that is an object's property. Constructor functions are invoked with *new*.

Within a method *this* is a reference to invoking object (*receiver*), not the defining object. In non-method functions *this* refers to the global object.

**Q:** What if I want to call function $f$ as a method of object $o$, but $f$ is not a method/property of $o$?
A: Use *call*, a method every function (including $f$) posesses. You can pass in your desired *reciever* object along with arguments.

**Q:** What if I want to extract a function's method and pass it to a higher order function?
A: Use bind to prevent the incorrect reciever prob

### 2.2 Callbacks

**Overview** Function$b$ is passed into function $a$ as an argument and $a$ executes $b$ at the end of its body. The main consideration in using callbacks is asynchronous-ness.

## 3 Closures

**Overview** Functions that store references to variables from their containing/enclosing scopes. State hidden within behavior.

Closures can update variables from enclosing scopes. Inner functions contain the scope of parent functions even if the parent function has returned.

## 4 Operators

**Q:** == vs === ?

## 5 Libraries

### 5.1 underscore.js

**Overview** A bunch of functions to make javascript more functiontional - provides map, reduce, filter, etc.

## 5.2  require.js

**Overview**   To write modular javascript while optimizing by reducing HTTP requests for js resources.

**Modules**   Are scoped so they don't pollute the global namespace. A good description of the Javascript module pattern at *http://www.adequatelygood.com/JavaScript-Module-Pattern-In-Depth.html*

Each module file is of the basic form define( STUFF ); Where STUFF parameters can include dependencies (other modules) and the definition of an object literal or function. RequireJS figures out all dependencies

## 5.3  prototype.js

## 5.4  backbone.js

**Overview**   Client-side MVC.

Data/State is represented as *Models*. When a model gets changed it fires a *change* event. All Views that display that model's state update themselves in response to the change.