Functional Programming with UnderscoreJS

- **Introduction**

Topics we'll cover today include:

- **A brief overview of Functional Programming**
- **What UnderscoreJS is and how it can help**
- **Examples of using higher-level programming such as each, map, reduce, include, any.**

What is functional programming?

- **JavaScript is a functional programming language**
  - **It's actually a multi-paradigm language supporting Object-oriented, imperative, and functional styles.**
- **Functional language concepts include:**
  - **First class functions – they can be passed as arguments**
  - **Support for anonymous functions**
  - **Ability for Functions to be assigned as variables**
  - **Support for nested functions**
  - **Support for closures**
- **What are the benefits functional programming**
  - **Emphasizes "What and Why" rather than "How"**
  - **Example:**
    - ***Procedural: k = 1; while k < 11; print k;***
    - ***Functional: countToTen => print (range(1,11) ;***

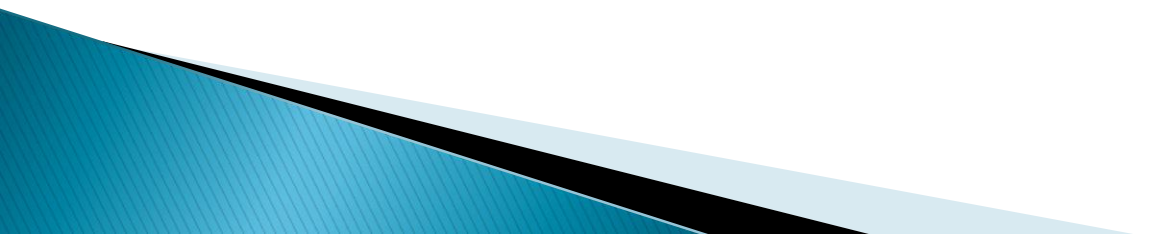
What is UnderscoreJS?

- **A utility-belt for functional programming with JavaScript**
  - **Brief introduction of UnderscoreJS utility functions**
    - ***UnderscoreJS provides about 60 utility functions to deal with Arrays, Collections, and Objects. All of which work very similarly. Once you master the basic functions the rest are straight forward.***
  - **Development team and contributions**
    - ***UnderscoreJS is an open source project hosted on github.***
    - ***Authored by Jeremy Ashkenas, DocumentCloud Inc.***

A note on style: Functional vs. Object oriented?

- **Choose a style that works for you and stick with it**
  - **Underscore allows both object and functional styles.**
  - **With the object style you can chain methods such as jQuery.**
  - **We'll focus on the functional style today.**

Higher-level programming



Iterating a list

- **Procedural:**
  - **for(i=0;i<myArray.length;i++){**
  - **print(i + “: “ + myArray[i])**
  - **};**
- **Functional with Underscore:**
  - **_.each(myArray, function(element, i){print (i+”: ” + element)});**
- **Notes: fewer lines, more descriptive, built in protection**

Mapping and Reducing

- **_.map(myArray, print);**
- **var total = _.reduce(myArray, function(memo, element){ return memo + element.price; }, 0);**

Data structures vs. nested conditionals

- **Nested Conditionals:**
  - **if (value === -1) return “no”;**
  - **else if (value === 0) return “maybe”;**
  - **else if (value === 1) return “yes”;**
- **Data structures:**
  - **var ruleMap = {**
  - **“-1”:”no”,**
  - **“0”: “maybe”,**
  - **“1”:”yes”};**
  - **return ruleMap[value];**

(Include / Any)

- **Nested Conditional**
  - **If (valueSelected === self.id() ||**
  - **valueSelected === ‘’ ||**
  - **valueSelected === undefined) {**
    - ***if (self.hasVisibleItems()) return true;***
  - **}**
  - **return false;**
- **Include / Any**
  - **return _.include([self.id(), '', undefined], valueSelected) && self.hasVisibleItems();**

UnderscoreJS in your applications

- **Where to get UnderscoreJS:**
  - http://documentcloud.github.com/underscore/#
- **Bootstrapping your project with Underscore**
  - **<script type="text/javascript" src=" underscore.js"></script>**
  - **The Module pattern**
    - ***Resource:*** *http://www.adequatelygood.com/2010/3/JavaScript-Module-Pattern-In-Depth*
    - ***var module = function(_){***
      - **var public = {},**
      - **privateClients = _.uniq(clients);**
      - **public.partnerList = _.without(partners, privateClients),**
      - **return public; }**
    - ***module.partnerList();***

Live demo of Underscore in action

- **Example code hosted on githib / UpFront-Underscore**
- **Run in chrome developer console**
- **Next steps**
  - **https://github.com/UpFront-Underscore/functional-programming-examples**
  - **http://www.ibm.com/developerworks/library/wa-javascript/index.html**
- **Q&A**