FUNCTIONAL PROGRAMMING WITH JAVASCRIPT

Abdul Habra @ahabra

Presented at
Great Lakes Functional Programming Conference 2012

Ice Breaker and Me

* I am incapable of making funny jokes, or even repeating funny ones in a funny way. So read these on your own and smile (or at least pretend)
  * Eight bytes walk into a bar. The bartender asks, Can I get you anything? Yeah, reply the bytes. Make us a double.
  * Why do programmers always mix up Halloween and Christmas? Because Oct 31 equals Dec 25.
* Me: Old timer, play with FP for past 2 years.

What You Should Expect

* Introduction
* Pure Functions
* Functions are Objects in JS
* Higher-Order Functions
* Lambdas (aka Anonymous Functions), Closures
* Combinators: map, filter, reduce (aka fold), reduceRight, forEach
* Curry / bind
* FP Libraries: underscore.js, Functional JavaScript
* Q/A: Ask whenever you want
* Bonus: How To Implement (if we have time)

What Do I Expect

* You know basics of JavaScript syntax
* You do not have to know functional programming
* You ask me only easy questions, and I say “That’s a great question”
* No questions about monads

Disclaimer

* JavaScript is not a perfect FP language
* No native support for immutability

Pure Functions

* Look at example 02-pureFunctions.js. JS syntax and semantics
* Have no side effects
* For same input, they return same output
* Examples: pi(), add(a,b), capitalize(s)
* Impure Functions: setters/getters, time(), updateDom(), I/O related functions.

Functions Are 1st Class Objects

* Look at example 03-functions.js and its spec
* Methods on function include: apply, call, toString, bind
* Functions can be passed as arguments to other functions

```
function add(a,b) { return a+b; }

add(1,2);  // returns 3

typeof add    // returns function
```

Higher-Order Functions

* Look at 04-hiOrderFunction.js and its spec
* Functions that takes functions as arguments
* Functions that return other functions
* Examples: functions that take callback arguments
* Can you suggest some?

Lambdas

a.k.a. Anonymous Functions

* Look at 05-lambda.js and its spec
* Notice how visitEach() takes a lambda argument and applies it on each element in the array

Closures

* Look at 06-closure.js and its spec
* Function that refer to variables defined outside the function in a scope surrounding the function
* The function “closes” around those variables to keep them available when the function is invoked.

Map

* Look at example 07-mapSpec.js
* Map here is a function !== Java Map
* map() is a function on the Array object
* map() processes each item in the array and “maps” it to a new value.
* It returns another array with **same size**, possibly with **different data types**
* map() can work on String objects

Filter

* Look at 08-filterSpec.js
* Returns a **subset** of an array for items which satisfy a condition

```
var positives= [1, 3, -5, 19, -36].
  filter(function(num) {
          return num >= 0; });

expect(positives).toEqual([1, 3, 19]);
```

Reduce
aka reduceLeft or fold

* Look at 09-reduceSpec.js
* Given a function that take 2 args, a and b, and return some new value based on the args
* Call the function on an initial value and 1st element in the array
* Call the function on result from last step and 2nd element in array
* and so on ...

ReduceRight
aka foldRight

* Look at 10-reduceRightSpec.js
* Similar to reduceLeft but starts from the end of the array

forEach

* Look at 11-forEachSpec.js
* Calls a function on each element in an array

Combinators

* The functions: map, reduce, filter are called combinators
* Example: Get the result of Filter, apply Map on it, then pass the result to Reduce (combine functions together)

Curry
aka bind

* Look at 12-currySpec.js
* Not related to Indian cuisine
* Named after Haskell Curry
* When a function takes several arguments, and you want to call it with some constant values for these arguments
* You can create another function that keeps the constant args and takes only the changing ones

FP Libraries

* Some of the functions (map, filter, ...) are not supported consistently on all browsers
* Libraries attempt to have uniformed syntax across browsers
* underscore.js: general purpose JS lib with several FP functions
* Functional JavaScript: More advanced
* Look at their home pages

References

* http://www.devtopics.com/best-programming-jokes/
* http://documentcloud.github.com/underscore
* http://osteele.com/sources/javascript/functional/
* JavaScript: The Good Parts, Douglas Crockford

Q/A

* If early, we can look at how to implement the combinators