FUNCTIONAL PROGRAMMING

JavaScript as development platform

Developed almost 20 years ago, JavaScript has outstanding expressiveness and flexibility allowing developer to choose how and what to program with JavaScript rather than dictating a way it meant to be used by its creators. Programming paradigm, style, naming are all aspects that make possible to bring the best from other languages to JavaScript and avoid common pitfalls in the language known for years. During this module you will be challenged with series of problems intended to familiarize you with well-known functional concepts in a context of JavaScript.

Problem 1: Partial Application

Implement function *F* that allows to do partial function application in a form of:

$$G(x, y, z \ldots) = N$$

$$F\big(x,\ G(x, y, z \ldots)\big) \rightarrow H(y, z \ldots) = N$$

F should accept any number of parameters to apply.
G may accept any number of parameters.

Is there any JavaScript built-in alternative?

In order to solve this problem you have to be familiar with next concepts:

1. Partial Application
2. High-order and First-class functions
3. Activation Object and handling of variable number of function arguments

Problem 2: Currying

Implement function *curry* that allows to do currying like:

$$f(x, y, z) = N,$$

$$curry(f) = x \rightarrow (y \rightarrow (z \rightarrow N))$$

Function f may accept any number of explicit parameters.
Implicit parameters are not subject to curry.

How is it differ from Partial Application?

In order to solve this problem you have to be familiar with next concepts:

1. Currying and Partial Application
2. High-order and First-class functions
3. Activation Object and handling of variable number of function arguments

Problem 3: Linear fold

Implement linear fold function that works on arrays:

F(array, callback[, initialValue]),

callback: Function to execute on each value in the array, taking four arguments:

previousValue: The value previously returned in the last invocation of the callback, or initialValue, if supplied.

currentValue: The current element being processed in the array.

index: The index of the current element being processed in the array.

array: The array fold was called upon.

initialValue: Object to use as the first argument to the first call of the callback.

Does ES5 has built-in alternative?

In order to solve this problem you have to be familiar with next concepts:

1. Folding and unfolding
2. High-order and First-class functions

Problem 4: Linear unfold

Implement linear unfold function that returns a sequence that contains the elements generated by the given computation:

F(callback, initialValue)

callback: A function that takes the current state and returns a tuple consisting of the next element of the sequence and the next state value. Callback accepts current value and produces new state and element. Unfold stops upon falsy value returned by callback.

initialValue: The initial state value.

In order to solve this problem you have to be familiar with next concepts:

1. Folding and unfolding
2. High-order and First-class functions

Problem 5: Map

Implement a function that creates new array with the results of calling a provided function on every element in this array.

Does ES5 has built-in alternative?

In order to solve this problem you have to be familiar with next concepts:

1. Functional Map
2. High-order and First-class functions

Problem 6: Filter

Implement a function that filters array based on callback result.

Does ES5 has built-in alternative?

In order to solve this problem you have to be familiar with next concepts:

1. High-order and First-class functions

Problem 7: Average of even numbers

Given array of numbers, find average of even ones using functions implemented for previous problems. Example: [1,23,2,6,12, 0] -> (2 + 6 + 12 + 0) / 4 = 5

In order to solve this problem you have to solve 1-6 problems.

Problem 8: Sum of random numbers

Get the sum of 10 random numbers using functions implemented for previous problems.

In order to solve this problem you have to solve 1-6 problems.

Problem 9: First

Implement a function that returns the first element in array that satisfies given condition.

In order to solve this problem you have to be familiar with next concepts:

1. High-order and First-class functions

Problem 10: Lazy evaluation

Implement a function that takes list of parameters and makes any given function lazy.

In order to solve this problem you have to be familiar with next concepts:

1. High-order and First-class functions

Problem 11: Memoization

Implement a function that for any given function F produces function G that caches its previous calling results.

Function F accept single explicit parameter.
Implicit parameters should not be taken into account.
Watch out for NaN, undefined and circular references.

In order to solve this problem you have to be familiar with next concepts:

1. Memoization
2. High-order and First-class functions