Unesp >< IGCE >< LPNC >< BCC Not 2016

Aula 1 – Roteiro

Livro texto para a Unidade I (4 aulas) : Professor Frisby’s Mostly Adequate Guide to Functional Programming

* Versão original em inglês: (<http://gitbook>...
* Versão traduzida para português: ...

O livro está disponível em diversos formatos.

Ferramenta para JavaScript: node.js

Node.js é “JavaScript no servidor”. Mais especificamente, é um sistema de programação “assíncrona” baseado em “call-backs”. É implementado com o intepretador JavaScript V8 do Google Chrome. Possui o gerenciador de pacotes NPM, que é muito popular entre programadores JavaScript em geral.

Node.js é usado na disciplina apenas como um interpretador JavaScript fora de um navegador, com facilidade de instalação de bibliotecas (NPM).

Instalação do Node.js : Instale o NVM (Node Version Manager). Qualquer versão do Node.js deve servir. Recomenda-se o sistema operacional Linux, mas outros sistemas não estão excluídos.   
Siga as instruções do NVM.

Para executar o Node.js na linha de comando do sistema operacional, digite node

IC1 em JavaScript

Experimente digitar expressões em JavaScript no interpretador e observar os resultados. Exemplos:

2+2

console.log(“Hello”)

var i = 10

i + 100

Digite o seguinte programa no arquivo jsdemo.js :

var i = 2;

for (i = 0; i < 10; i++) {

console.log(i+100);

}

Carregue o arquivo no Node.js: node jsdemo.js

Capítulo 2 : Funções de Primeira Classe

A expressão “funções como cidadãs de primeira classe na linguagem de programação” indica que funções podem ser armazenadas como valor em uma variável ou array assim como os números, as strings, etc. Mais que isso: funções podem ser valores de parâmetros de um função ou o seu valor de retorno.

Há uma forma mais rigorosa (dentro da Matemática) de se referir a esse conceito: funções de ordem superior, pois elas aceitam funções como parâmetros e podem ter funções como resultado. Ou seja, são funções cujo domínio e cuja imagem são o conjunto das funções.

Essa característica do JavaScript foi influenciada pelo Lisp, e é essencial para a moderna concepção funcional do JavaScript.

var a = function(x) {

return x \* 2;

}

A chave é entender que a expressão function(...) {...} em JavaScript representa uma função como “valor”. Tal expressão é, às vezes, chamada de “função anônima”. Em Lisp, é conhecida como “lambda”, pois remete ao formalismo lógico conhecido como “Cálculo Lambda”, equivalente às máquinas de Turing.

Siga o Capítulo 2 do livro texto até a sua segunda página pelo menos.

Capítulo 4 : *Currying* (“Curryficação”?)

Esse princípio foi estabelecido nos anos 1920 por Moses Schönfinkel e Haskell Curry. O termo foi designado em homenagem ao segundo.

Em resumo: para qualquer função, é possível criar uma função equivalente que aceita um só parâmetro, e retorna uma função que espera os outros parâmetros.

var soma = function (x) {

return function (y) {

return x + y;

}

}

soma(10)(100)

var soma20 = soma(20)

soma20(5)

É possível criar uma função de ordem superior que, recebendo uma função qualquer, devolve sua versão “currificada”. A biblioteca ramda contém uma função como essa, chamada curry .

Obtendo e usando bibliotecas no Node.js

Na linha de comando do sistema operacional, no diretório de trabalho, digite   
npm install ramda

O NPM traz a biblioteca pela internet e a instala no subdiretório node\_modules .

Inicie o Node.js no diretório de trabalho. Dentro do node, digite

var curry = require(“ramda”).curry;

Em seguida, use a função curry:

var max = function (a,b) {

if (a>b)

return a;

else

return b;

}

var max\_c = curry(max);

var vinte\_ou = max\_c(20);

vinte\_ou(30);

vinte\_ou(10);

De volta ao conceito de *Currying*

*Currying* é o processo de conversão de uma função.

*Aplicação Parcial* é o processo de empregar poucos argumentos à função “curryficada”.