Resenha da palestra “Learning Functional Programming with JavaScript”

Disciplina: Paradigmas Linguagem de Programação
Curso: Ciência da Computação - UniCEUB
Período: 5º semestre, 2021-2
Palestrante: Anjana Vakil

Alunos:
Danilo Rebouças Fernandes de Lima **RA:** 21953484
Hendrik Rodrigues Melo **RA:** 21952356

A palestra “Learning Functional Programming with JavaScript” aborda o tema de Linguagem Funcional. Anjana Vakil, a palestrante, conta que no outono de 2015 foi para uma comunidade de programação em Nova Iorque chamada "The recurse center", trata-se de um retiro de programação onde você pode ir para aprender sobre qualquer coisa que você acha interessante no mundo da programação. Foi aí que teve o seu primeiro contato com a linguagem JavaScript. Ao longo da palestra ela aborda conhecimentos e conceitos de linguagem funcional, explicando suas definições, usos, aplicações e funcionalidades.

São apresentados alguns conceitos específicos como os de função pura, *map*, funções de ordem superior e conhecimentos sobre linguagem funcional. São exploradas abordagens como uso de dados imutáveis.

Função pura é uma em que o seu retorno depende apenas de suas entradas, não utilizando, por exemplo, variáveis globais. A sua saída não é alterada se outras variáveis ou partes do código forem modificadas. Um erro de saída em uma função pura é mais fácil de ser depurado uma vez que basta checar o valor de entrada e a própria função. Se a entrada em uma função pura for a mesma, o seu retorno será o mesmo.

São também exploradas algumas práticas de linguagem funcional como a imutabilidade e o uso de funções em detrimento de objetos e classes.

Com a imutabilidade, ao invés de alterar o valor de uma variável, é criada uma outra variável com o novo valor desejável. Por exemplo, se temos uma lista `var planetas = ["Terra", "Marte", "Plutão"];`, se desejarmos retirar o plutão da lista esse processo é realizado criando uma nova lista sem esse elemento, mantendo a primeira com os valores originais. Essa prática evita erros durante a execução do código por alteração de valor ou tipo de uma variável.

Por último, fala sobre o uso de árvores binárias para otimizar processos em listas, fazendo melhor uso de memória e aumentando a velocidade de execução.
Como exemplo, ela cita uma execução em que com a árvore binária basta alterar apenas um nó para gerar o mesmo resultado, enquanto que sem ela é necessário retornar uma lista completa.

A palestrante explica sobre o uso de *map*, permitindo que menos loops iterativos como for e while sejam usados no código e possibilitando a aplicação de funções em uma lista. O *map* permite a aplicação de uma função em todos os elementos de uma lista.
Os conceitos de *filter* e *reduce* são citados apenas.

As funções de ordem superior consistem no uso de funções em que uma ou mais funções são retornadas na saída ou são usadas como entrada. Permite o uso de métodos recursivos e em cascata.

O uso de classes pode, como dito pela palestrante, criar situações embaraçosas a partir do uso de conceitos relativamente complexos como herança e polimorfismo.

Essas práticas tem como objetivo evitar erros, problemas e facilitar o debug do código e o teste de programas, bem como favorecer a checagem formal de erros. Utilizando, por exemplo, variáveis imutáveis é mais fácil garantir que a variável tenha o valor desejado, que não seja utilizada com o tipo ou valor incorreto ou que seu valor seja alterado por uma linha do código difícil de ser encontrada. O uso de *map* facilita o entendimento do código sobretudo quando é extenso. Uma função dobra(lista), que dobra os valores de elementos de uma lista, é de fácil compreensão. O mesmo processo com iteração pode ser descrito da seguinte forma:

```
# código em Python
for i in range(len(lista)):
    lista[i] = 2 * lista[i]
```

Desse modo, não é tão clara a intenção do programador de dobrar os valores da lista quanto em dobra(lista). Um código de fácil compreensão facilita a manutenção e o uso dele por outros usuários ou comunidade, reduzindo o tempo gasto consertando erros.

As árvores binárias tem importantes vantagens como diminuir o tempo de busca. O retorno de um elemento específico de uma lista demanda tempo de O(n), mas com árvores binárias é reduzido para O(ln(n)). A alteração de um ou mais valores pode ser feita sem armazenar toda a lista e seus elementos simultaneamente na memória ao se utilizar apenas os nós da árvore binária.

Dessa forma, a palestra aborda o tema de linguagem funcional explicando o porquê de ter sido criada, suas aplicações, seus principais conceitos e conhecimentos além de citar outras fontes de leitura complementares sobre o tema, ampliando a compreensão do ouvinte sobre o tema.