Aula 45 Exercícios da tarde

* Exercício 1

Hoje conhecemos mais profundamente o node.js, e vimos o tanto de coisa que dá para fazer com ele.

Uma característica muito interessante do node.js é que ele é rodado pelo terminal, e pelo terminal podemos passar parâmetros para o nosso código. Os parâmetros do node funcionam como um array, então seu primeiro valor está no índice 0. Um ponto muito importante é que o node já vem com os dois primeiros parâmetros preenchidos.

a) Como fazemos para acessar os parâmetros passados na linha de comando para o node?

b) Crie um programa que receba seu nome e sua idade. Após receber estes valores, imprima no console uma mensagem que siga a seguinte estrutura:

"Olá, (Nome)! Você tem (sua idade) anos."

c) Altere o programa acima para que mostre também a sua idade daqui a sete anos.

"Olá, (Nome)! Você tem (sua idade) anos. Em sete anos você terá (nova idade)"

* + Dica

Os parâmetros do node sempre chegam como strings. Para podermos fazer cálculos, precisamos transformar o valor em um number.

const valor = Number(valorEmString)

* Exercício 2

Crie uma aplicação Node que recebe uma string representando uma operação matemática e dois valores numéricos. O retorno deverá ser o resultado da operação selecionada utilizando os 2 valores fornecidos.

* + **Exemplos**
  + npm run start add 2 2
  + // "Resposta: 4"
  + npm run start sub 10 2
  + // "Resposta: 8"
  + npm run start mult 50 2
  + // "Resposta: 100"
  + npm run start div 100 2
  + // "Resposta: 50"
  + **Dicas** (clique aqui se travar em algum pedaço)
    - Lembre-se da ordem de recebimento de argumentos do node. Para acessar os argumentos, usamos o objeto process.argv, que é um array de todos os parâmetros usados pelo node. Os argumentos [0] e [1] são fixos, onde o primeiro é o próprio node e o segundo é o arquivo que está rodando. Todos os valores são armazenados como string.
  + process.argv[0]: string = //node
  + process.argv[1]: string = //o arquivo
  + process.argv[2]: string = //primeiro argumento
  + process.argv[3]: string = //segundo argumento
    - Você pode separar o tipo de operação usando um switch para avaliar o tipo de operação passada como parâmetro para o node.
  + switch(operacao){
  + case "soma":
  + //seu código aqui
  + break;
  + case "subt":
  + //seu código aqui
  + break;
  + }
* Exercício 3

Crie uma aplicação Node que receba uma string representando o **nome do arquivo da lista de tarefas** e uma string representando **uma nova tarefa**, em seguida o programa deve adicionar a **nova tarefa** em um arquivo que tenha **o nome da lista de tarefas.** Para isso, crie um arquivo chamado tarefas.txt

* + Exemplos
    - Checando conteúdo de um arquivo de texto nomeado tarefas.txt
  + $ cat tarefas.txt
  + Comprar Pão
  + Comprar Sabonete
    - Executando nossa aplicação, fornecendo a lista usada acima como primeiro parâmetro.
  + $ npm start tarefas.txt "Comprar Leite"
  + Tarefa adicionada com sucesso!
    1. Checando o conteúdo da lista de tarefas novamente
  + cat tarefas.txt
  + Comprar Pão
  + Comprar Sabonete
  + Comprar Leite
  + **Dicas**

Lembre-se que o nome do arquivo que vamos alterar (o tarefas.txt) também precisa ser passado como parâmetro do node, e precisa estar na mesma pasta do arquivo TS a ser executado.

Para apenas adicionar informações de texto a um arquivo, o gerenciador de arquivos do node, o nosso querido fs, tem uma função chamada **appendFileSync**. Esta função, diferente do writeFileSync que usamos em aula, apenas adiciona informações, sem se importar com o conteúdo já existente no arquivo.

fs.appendFileSync("nome do meu arquivo.txt", "texto a ser adicionado");

* Exercício 4

Transforme os dois exercícios anteriores em projetos individuais do node (crie uma pasta para cada um)

* + Requisitos

O projeto deve:

* + - Conter um package.json
    - Ter um script start configurado e funcionando.

Para isso, recomendo que você dê uma olhada na seção "**Caso eu queira começar o projeto do zero, o que deve ser feito**", para lembrar de como usamos o npm init, e como configuramos o script start do projeto.

* **Desafios**
  + Ainda nos exercícios 1 e 2 adicione verificações para garantir que os parâmetros passados estão corretos e informe ao usuário caso não estejam. Exemplo: "Esperava 2 parâmetros só recebi um."
  + Volte nos exercícios 1 e 2 e faça com que o texto impresso no terminal (usandoconsole.log) seja colorido.