Função nameFormated

Essa função tem como objetivo deixar os nomes dos produtos escritos corretamente, corrigindo certos caracteres.

Primeiro eu recebo o nome de cada produto através do parâmetro. Após isso, fiz duas listas : uma com os caracteres considerados errados e outro com os caracteres que deverão ser colocados no lugar. É importante ressaltar que a posição de cada letra é igual, ou seja, o " ç" está na mesma posição que o "c".

Depois disso, divido, através do método split, cada letra do nome do produto e adiciono em um array. Chamo uma função para cada letra do nome, e verifico se tem algum igual no array que estão as letras erradas. Se tiver, ele irá trocar pela letra que está na mesma posição no array de lettterFormated(array que ficam as letras certas) e por fim, irá retornar.

No final, apenas junto todos os caracteres através do método .join()

Função quantityFormated

Essa função tem como objetivo verificar se a quantidade do produto está correto e se existe essa propriedade.

Primeiro eu recebo a quantidade de cada produto. E com uma condicional verifico se é inexistente ou sem valor. Se a condição for verdadeira, adicionamos essa propriedade atribuindo um valor 0 para ela. Se não, ele mantém o valor atual.

Função priceFormated

Essa função tem como objetivo deixar todos os preços no tipo number.

Recebo o preço de todos os produtos e retorno o preço corrigido através da função parseFloat, que tem por objetivo transformar string em ponto flutuante.

Função sendValues

Tem por objetivo enviar os valores formatados para um arquivo .json

Usando o módulo fs junto com a função writeFile, estou enviando os novos valores(formatados) convertidos para JSON para o arquivo saida.json.

Depois uso uma condicional para verificar se houve erros. Se não houve erros, iremos chamar as funções de testes(list e calculatorPriceOfCategory) passando os valores que acabaram de ser escritos no saída.json

Função list

Tem como objetivo imprimir a lista com todos os nomes dos produtos, ordenados primeiro por categoria em ordem alfabética e ordenados por id em ordem crescente.

Usando o método sort organizei os valores passados como parâmetro. As condições verificam qual categoria vem antes na ordem alfabética, e no último caso, se as categorias forem iguais, verificamos qual tem o id menor.

Após isso, apenas adicionei cada nome de produto em um array. E por fim, mostrei no console cada nome.

Função calculatorPriceOfCategory

Tem como objetivo calcular qual é o valor total do estoque por categoria.

Começo criando um objeto que receberá os valores totais por categoria. Usando o array com os dados corrigidos, que foi recebido no parâmetro, executo o método forEach para verificar todas as posições do array. Com o auxílio do nome da categoria(element.category) vejo se existe essa chave no objeto criado. Se não existir, significa que o elemento atual do forEach será o primeiro nessa categoria, então será adicionado o seu preço multiplicado pela quantidade.

Se caso já existir, multiplicaremos o preço pela quantidade e somaremos esse resultado ao valor que já existe nessa categoria.