Documentação do código resolução.js

Este código foi desenvolvido para arrumar o código do broken-database.json que foi corrompido devido a um bug.

AbriArquivoJson ()

Neste código por primeiro de tudo foi criado uma primeira função da qual chamei de AbrirArquivoJson que serviu para ler o arquivo.json e poder utilizá-lo no Javascript, para isso utilizei do método de leitura 'fs' onde o .readfile serve para ler o arquivo, nos parênteses é passada a localização do arquivo e em seguida um 'utf-8' que seria o encoding dessa aplicação. O JSON.parse serve para passar os arrays de 'strings' para objetos.

ArrumarNomes ()

Esta segunda função criada foi feita para corrigir o bug que quebrou algumas letras dos nomes no arquivo, nela foi utilizado um for para englobar todos os itens do arquivo .json, e depois para substituir as letras quebradas pelas corretas foi utilizado o .replace assim trocando e consertando as nomenclaturas.

ArrumarPrecos ()

Na terceira função criada foi feito um tratamento de código para transformar todas as informações no atributo price para numbers, pois com o corrompimento do arquivo alguns ficaram como strings, para isso também utilizamos um for para englobar todos os itens do arquivo, e depois pegando especificamente este atributo price utilizamos o Number.parseFloat forçando a todos eles virarem numbers assim passando os que estavam em string de volta para numbers.

ArrumarQtd ()

A quarta função criada foi feita para tratar quantidades que com o bug foram removidas de alguns itens, essas quantidades removidas tinham o valor de zero por isso acabaram sendo tiradas, para resolver e recolocar essas quantidades, utilizei novamente o for para englobar todo o arquivo e depois foi colocado o if para dizer que se o atributo quantity não existisse ou fosse igual a 'undefined' então o sistema deveria colocar por padrão a quantidade zero naquele estoque.

Essas duas funções criadas abaixo tem o intuito de validar a recuperação do arquivo original antes de ser corrompido.

OrdenarEPrintar ()

Está função for criada para ordenar os nomes dos produtos primeiro por categoria em ordem alfabética e depois ordenar por id em ordem crescente. Para isso utilizei do método .localeCompare que funciona comparando as strings para ver quem vem primeiro, no caso dessa aplicação comparei os atributos category assim colocando eles em ordem alfabéticas.

CalculaEstoquePorCategoria ()

Esta segunda função com a finalidade de validar as funções feitas acima consertando o banco de dados do arquivo tem por objetivo calcular o valor em estoque de cada item. Para isso primeiro criei uma variável categorias = {} para onde os valores de cada estoque irão, por segundo coloquei um for para englobar todos os elementos, depois foi colocado um if para dizer que se o código a frente for verdadeiro para executar a linha de baixo. Dentro desse if coloquei um "(String(categorias[jsonArray[i].category]) === 'undefined')" onde basicamente o que diz é se a variável categorias for igual a 'undefined' vai executar o código na linha de baixo que diz para setar a variável categorias para 0, a string na frente do código serve para força-lo a ser string. Depois disso para calcular o valor do estoque de cada categoria fiz a variável categoria += o preço vezes a quantidade para obter o valor total de cada estoque.

Let dados

Criei o let dados para ficar mais organizado e simples de exibir as informações passadas pelas funções criadas acima, basicamente na primeira parte ele funciona somando as funções ele grava uma função na variável e logo em baixo salva a próxima função e assim sucessivamente, depois adicionei um console.log de texto para ficar mais claro o que seria exibido e no console.log de baixo adicionei a variável onde estava todas as informações, na parte de ordenar a lista e a mesma lógica, porém, puxando a função OrdenarEPrintar.

Método de saída

Este método é utilizado para copiar o código exibido no console e criar um arquivo .json atualizado com todo o banco de dados corretos. "O fs.writefile" serve para indicar que você quer reescrever o código criado, a "saída.json" é o nome do arquivo que será criado e exibira o banco de dados atualizado, "JSON.stringfy(dados,null,2)" é para transformar os itens de objetos para Strings e também para formatar a forma que o código vai ser impresso no próximo arquivo. O encoding adicionado abaixo serve para informar que o padrão seria "utf-8" e o flag "w" e para sempre que o código for alterado o arquivo atual é só atualizado e não é necessário a criação de um novo. Em

baixo são mensagens informativas para caso ter dado erro, e também para quando ter dado certo.

Escolha da linguagem

O código citado acima foi desenvolvido na linguagem JavaScript, escolhi essa linguagem, pois lendo o documento em que é passado as informações de como desenvolver o projeto foi a qual eu entendi que deveria ser feito.

Tratamentos para evitar bugs

Para evitar os futuros bugs na aplicação foi desenvolvido alguns “try, catch”, e “if” no código, o qual servirão para que se tenha algo errado que poderá quebrar o código ele vai aparecer uma mensagem de erro falando o que está errado para facilitar a manutenção de quem está mexendo, como, por exemplo na função AbrirArquivoJson, que coloquei informações essenciais para o código ser rodado, caso essas informações já não cumpram estas regras já vai dar um erro de cara, assim não sendo necessário executar o código todo para depois perceber que tem algum erro.

Conclusão final

Foi um código muito legal de se desenvolver e que me proporcionou bastante aprendizado já que nunca tive contato com bastante das coisas que precisaram ser utilizadas para fazê-lo funcionar, então depois da conclusão desse projeto posso dizer que aprendi coisas novas que poderão ser utilizadas daqui para frente, desde já agradeço por essa oportunidade e tomara que atinja as expectativas criadas para o código.