Correção de dados de um arquivo JSON

A linguagem utilizada foi Javascript, é a linguagem que estou mais habituado e venho estudando, por isso, a escolha

Índice

1. **Figura 1.**
   Função para corrigir os nomes com caracteres especiais

```
const fs = require('fs/promises')
const data = require('./broken-database.json')

function changeNames(nameProduct){

    const productName =  nameProduct.map((product)⇒{

       const namesProduct =  product.name

       const fixNames = namesProduct.replace(/[æ]/g,'a')
       .replace(/[¢]/g,'c')
       .replace(/[ø]/g,'o')
       .replace(/[ß]/g,'b')



       return fixNames
    })

    const jsonCorrectNames = data.map(getName⇒ getName.name = productName.shift())

    return jsonCorrectNames

}
```

Primeiro foi feita a leitura do arquivo Json e armazenada em uma constante que retorna um objeto, e a utilização do modulo fs do nodeJS para lidar com os arquivos locais do sistema.

Em seguida, a função changeNames que recebe o arquivo json como parâmetro, onde foi criado um map pra retornar somente o nome do produto onde é armazenado na variável namesProduct. Após, foi criada outra variável chamda fixNames, onde recebe namesProduct com a correção dos caracteres especiais sendo substituído pelo correto utilizando um aninhamento de replaces() e retornado todo os nomes corrigidos.

Após isso, foi feito um segundo map que recebe diretamente o arquivo json de fora da função pra ser alterado em memória, onde foi “capturado” o nome do produto e passado o retorno da função com os nomes certos e utilizando o método shift para decrementar o array productName e assim ir colocando um de cada vez em cada laço

2. **Figura 2.**
   Função para corrigir o preço em String para Number

```
function stringToNumber(productsPriceString) {

    const productPrice = productsPriceString.map((productsPrice)⇒{

        const price = productsPrice.price

        return Number(price)

    })

    const jsonCorrectPrice = data.map(getPrice⇒ getPrice.price = productPrice.shift())

    return jsonCorrectPrice
}

changeNames(data)
stringToNumber(data)
```

Nesta segunda função, a lógica é a mesma da **figura 1**, percorremos da mesma forma a variável data na função stringToNumber, e no lugar do método replace, apenas foi retornado o price já convertendo pra Number, e posteriormente, substituindo em jsonCorrectPrice o preço antigo em string pelo preço correto em number e decrementando com shift também.

3. **Figura 3.**
   Função para corrigir a quantidade

```
function corrigeQtde(data){
    const qtde = data.map(item ⇒ {

        if(item.hasOwnProperty('quantity') === false){
           return item['quantity'] = 0
        }else{
            return item.quantity
        }

    })

    return qtde

}

changeNames(data)
stringToNumber(data)
corrigeQtde(data)
```

Nesta função, cada item(product) está sendo percorrido e verificando em cada passada se cada produto contém o atributo “Quantity” utilizando hasOwnProperty, e caso não tiver, irá retornar a adição desta mesma propriedade no item, caso o produto já conter o atributo “Quantity”, apenas retorna ele mesmo, e feito todo essa verificação, a função irá retornar todo o objeto com as devidas modificações feitas.

4. **Figura 4.**
   Criando um novo arquivo Json com as correções feitas

```
const newDataBase = JSON.stringify(data)

function  promise() {
    return fs.writeFile('./saida.json', newDataBase)
}
```

O método “writeFile” do Noje, espera receber uma string pra poder escrever no arquivo, sendo assim, foi convertido o objeto json no qual passou por todas as funções anteriores, para uma string e armazenada na variável newDataBase, que passamos como parâmetro para o método, lembrando que é uma promise, veremos o porquê mais a frente

5. **Figura 5.**
   Verificando a existência do arquivo

```
async function  validate() {
    await promise()

    const saidaJson = require('./saida.json')

    console.log('---'.repeat(30))

    console.log("Filtro por categoria e Id\n")
    console.log(FilterCategoryAndId(saidaJson))

    console.log('---'.repeat(30))

    console.log("Quantidade por categoria\n")
    console.log(FilterQuantityByCategory(saidaJson))
}

validate()
```

Como citado anteriormente, como criamos uma função chamada promise que é uma promise, chamamos ela na função validate que só então quando o arquivo for criado, o resto da função que depende desse novo arquivo, será executado, as funções executadas no validade será explicada mais a frente

6. **Figura 6.**
   Filtrando por categoria e Id

```
const FilterCategoryAndId = (json)⇒{

    const sortByCategoryAndId = (a,b)⇒{
        if(a.category < b.category){return -1}
        if(a.category> b.category){return 1}
        if(a.id < b.id){return -1}
        if(a.id > b.id){return 1}

        return 0
    }

    const result =  json.sort(sortByCategoryAndId)

      return result
  }
```

Como o método sort compara o conteúdo utilizando o Unicode UTF-16, que seria o código da palavra, podemos fazer essa comparação em uma outra função, no caso sortByCategoryAndId e passamos ela como parâmetro para o sort(), pois o sort() espera receber um parâmetro a e b para fazer a comparação, caso retornado -1, ele colocara na frente o elemento ‘a’, caso retornar 1, ele irá para a segunda posição, e caso for 0, se manterá na mesma.

7. **Figura 7.**
   Quantidade total de produtos por categoria

```
const FilterQuantityByCategory = (json)⇒{
  const categories = []

   json.map((product) ⇒ {
      const index = categories.findIndex((category) ⇒ category.category === product.category)


      const object = {
          category: product.category,
          quantityTotal: product.quantity * product.price
      }

      return index === -1
          ? categories.push(object)
          : categories[index].quantityTotal += product.quantity*product.price
  })

  return categories
}
```

Nesta função, foi criada um novo array pra armazenar o novo objeto contendo a categoria e quantidade total, foi mapeado e feita a comparação, se for retornado -1, que é não encontrado, será adicionado no objeto, caso contrário, se já existir, ele irá usar ele mesmo para ir concatenando e fazendo o cálculo

8. Referências

Conteúdos utilizados para pesquisa e desenvolvimento:

https://www.youtube.com/watch?v=tNCg3BGtG1w&ab_channel=EddieJaoude

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/shifthttps://dmitripavlutin.com/check-if-object-has-property-javascript/

Códigos utilizados:

Código usado do vídeo no minuto 4:41
https://www.youtube.com/watch?v=psZ_nDAoCFc&ab_channel=AllThingsJavaScript%2CLLC

Código usado do vídeo no minuto 2:14

https://www.youtube.com/watch?v=5hift2w4t-Q&ab_channel=JuniorDeveloperCentral

https://pt.stackoverflow.com/questions/124754/retirar-caracteres-especial-e-acentos-em-javascript

https://stackoverflow.com/questions/6712034/sort-array-by-firstname-alphabetically-in-javascript

9. Agradecimentos:

Muito obrigado por terem me dado esta oportunidade, pois independente do resultado, eu sei que eu subi um nível a mais, tive muitos desafios no processo e isso foi gratificante!

Gostou? então entre em contato :D

Contatar