Projeto Classificatório

Processo seletivo - Web Development

Candidato: Gusthavo Rangel Vieira

1. RECUPERANDO O BANCO DE DADOS

1.1 Ler o arquivo JSON

Para ler o arquivo “broken-database.json” criei a função *readFile()*, ela executa a função *readFyleSync()* levando como parâmetro o arquivo *.json*, que está no mesmo diretória do “resolução.js”. Após isso atribui e transformei em array os dados do arquivo *.json* à variável *data*.

```
function readFile() {
  var fs = require('fs')
  // Leitura do arquivo JSON
  var readFile = fs.readFileSync('./broken-database.json', 'utf8')
  // Transforma os dados do arquivo em array
  var data = JSON.parse(readFile)

  return data
}
```

Como seria utilizado muitas vezes os dados do arquivo *.json,* atribui a função *readFile()* à variável *dataBase.*

```
let dataBase = readFile()
```

1.2 Caracteres modificados

Utilizei os métodos *split()* e join() para separar a string pelos caracteres corrompidos e depois para juntar novamente com os caracteres correto.

```
//Função para substituir os caracteres "æ" por "a", "¢" por "c", "ø" por "o", "ß" por "b".
function replaceCharacter(names) {
  let replace = ''

  replace = names
    .split('ø')
    .join('o')
    .split('æ')
    .join('a')
    .split('¢')
    .join('c')
    .split('ß')
    .join('b')

  return replace
}
```

Levado como valor do parâmetro da função *replaceCharacter(),* o valor do parâmetro *name* da variável *dataBase.*

```
dataBase[position].name = replaceCharacter(dataBase[position].name)
```

1.3 String to Number

A função *fixPrice()* possui um condicional para verificar se o preço é do tipo string ou não, caso for verdade muda para o tipo Number.

```
//Função para corrigir preços do tipo String para Number
function fixPrice(prices) {
  if (typeof prices === 'string') {
    return Number(prices)
  }

  return prices
}
```

Como parâmetro é levado o *price* da variável *dataBase.*

```
//Corrigir preço
dataBase[position].price = fixPrice(dataBase[position].price)
```

1.4 Adicionar quantity

Para adicionar o quantity, fiz um condicional para verificar se existia o parâmetro *“quantity”* em cada objeto de cada posição do vetor. Caso não existisse ele retorna “undefined” e chama a função *addQuantity()* para adiciona-lo.

```
//Adicionar quantity
if (dataBase[position].quantity === undefined) {
  dataBase[position] = addQuantity(dataBase[position])
}
```

```
//Função para adicionar o quantity
function addQuantity(object) {
  object.quantity = 0

  return object
}
```

1.5 Salvando o novo JSON

Para criar o arquivo *"saida.json"* utilizei a mesma ideia para ler o arquivo, porém invés de utilizar o *readFileSync()* utilizei o *writeFileSync().*

```
function saveNewDataBase(recoveredDataBase) {
  var fs = require('fs')

  // Leitura do arquivo JSON
  var data = JSON.stringify(recoveredDataBase, null, 4)

  fs.writeFileSync('./saida.json', data)
}

saveNewDataBase(dataBase)
```

2. VALIDAÇÃO DO BANCO DE DADOS

2.1 Ordenando por categoria em ordem alfabética e por id e, ordem crescente

Através de uma função de validação com parâmetros (a,b), foi feita a ordenação por categoria e depois por id.

```
dataBase.sort(function (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
})

//Imprimindo os produtos por categoria em ordem alfabética e por id em ordem crescente
console.log('\n' + 'Produtos:')

for (let position in dataBase) {
  console.log('>' + JSON.stringify(dataBase[position].name))
}
```

2.2 Total em estoque por categoria

Para identificar as categorias criei uma função para buscar a categoria, e através do if e else validei as informações.

```
function searchPosition(vector, name) {
  var address = -1
  //localizar a posição do elemento passado como parametro
  address = vector
    .map(function (e) {
      return e.category
    })
    .indexOf(name)

  return address
}
```

```
//Somando o valor de estoque de cada categoria
for (let position in dataBase) {
  address = searchPosition(categories, dataBase[position].category)
  if (address >= 0) {
    categories[address].price +=
      dataBase[position].price * dataBase[position].quantity
  } else {
    categories.push({
      category: dataBase[position].category,
      price: dataBase[position].price * dataBase[position].quantity
    })
  }
}
```

3. POR QUE JAVASCRIPT?

Nessas duas semanas estudando Javascript percebi que é uma linguagem interessante para editar, criar e validar arquivos .json. Além disso a linguagem possui muitas facilidades como por exemplo ser fracamente tipado.

Outro ponto é que o Javascript possui uma versatilidade de frameworks, assim podemos utilizar a linguagem para web, mobile, banco de dados, etc.