

Documentação do código 'resolucao.js'

```
const fs = require('fs');
const bdBroken = require('./broken_database.json');
```

Nessa parte inicial, estou importando o módulo file system do node.js atribuindo à variável 'fs' para poder trabalhar com arquivos, e na variável 'bdBroken' estou atribuindo a importação do arquivo JSON corrompido.

```
function fixName(){
  bdBroken.forEach(registry => {
    registry.name = registry.name.replace(/æ/gi, 'a')
    .replace(/ø/gi, "o")
    .replace(/ç/gi, "c")
    .replace(/ß/gi, "b")}
  })
}
```

Aqui temos a função 'fixName', que com o forEach ela itera pelo JSON 'bdBroken' pela chave 'name', assim fazendo o replace das letras, quais estavam corrompidas.

```
function fixPrice(){
  bdBroken.forEach(registro => {
    registro.price = parseFloat(registro.price)
  })
}
```

Aqui temos a função 'fixPrice', que com o forEach ela itera pelo JSON 'bdBroken' pela chave 'price', transformando as strings em float.

```
function checkQuantity(){
  bdBroken.forEach(registro => {
    if(!registro.hasOwnProperty("quantity")){
      registro.quantity = 0;
    }
  })
}
```

Aqui temos a função 'checkQuantity', que com o forEach ele itera pelo JSON 'bdBroken' pela chave 'quantity', verificando se há uma chave com este nome, quando o if retorna false, quer dizer que ele não encontrou essa propriedade, então ele adiciona essa propriedade com valor 0.

```
function saveJSON(jsontoString){
  fs.writeFile("saida.json", jsontoString, function(err) {
    if (err) {
      console.log(err);
    }else{
      sortJSON();
      quotePrice();
    }
  });
}
```

Aqui temos a função 'saveJSON', onde com o módulo ele cria o novo arquivo JSON consertado, nos parâmetros são passados o nome que o novo arquivo terá('saida.json'), no próximo parâmetro é o conteúdo das alterações, e por último uma função callback que retorna um erro, caso alguma coisa dê errado, e no else temos as funções que serão explicadas mais à frente.

```
const bdNew = require('./saida.json');

function sortJSON(){
  bdNew.sort((item1, item2) => {
    return item1.id - item2.id
  });

  bdNew.sort((item1, item2) => {
    if (item1.category < item2.category) {
      return -1
    } else {return 1} return 0 })

  bdNew.forEach(registro => {
    console.log("|",registro.category,"|",registro.id,"|", registro.name)
  })
}
```

Primeiro, temos a importação do novo arquivo atribuído a variável 'bdNew', e depois temos a função 'sortJSON', que organiza o novo arquivo JSON consertado de forma que as categorias estejam em ordem alfabética e os produtos organizados pela id de forma crescente.

```

function quotePrice(){
  bdNew.sort((item1, item2) => {
    if (item1.category < item2.category) {
      return -1
    } else {return 1}
    return 0 })

  var categorias = bdNew.map((item) => {
    let category = {
      name: item.category,
      valor: 0}
    return category})
    .filter((set => f => !set.has(f.name) && set.add(f.name))(new Set));

  categorias.forEach((categoria) => {
    bdNew.filter(item => item.category == categoria.name)
      .reduce((acc, item) => {
        return categoria.valor = acc + (item.price*item.quantity);
      }, 0)
  })

  categorias.forEach(item => {
    console.log('|CATEGORIA|', item.name, '|VALORES| R$' , item.valor.toLocaleString('pt-BR'))
  })
}

```

Essa de longe foi a função mais complicada de fazer, pois não entendia sobre map(), filter() e reduce(), tanto que pedi ajuda à amigos e também busquei alguns códigos na internet.

No primeiro escopo de código dentro da função, o que eu faço é simplesmente ordenar os objetos pela sua categoria de forma alfabética, como foi feita na função anterior.

No segundo escopo de código onde crio a variável 'categorias', eu atribuo a ela um array construído pelo método map() a partir do 'bdNew', onde ele cria um objeto que recebe uma propriedade 'name', que adota o valor das 'category' dos objetos de 'bdNew', e também atribui a propriedade 'valor' recebendo como valor 0, porém só com o map() ele retorna um vários objetos com propriedade 'name' repetido. Para resolver isso, temos o filter(), que foi um código pego da internet([código filter](#)), que faz com que retorne um array com objetos sem repetir.

No terceiro escopo de código, aqui nós iteramos com forEach pelo array de objetos atribuídos a 'categorias', filtramos com filter() o 'bdNew' comparando o nome das categorias de 'bdNew' com 'categorias', para que o reduce() atribua ao categoria.valor o valor certo de cada categoria.

Por fim, itera-se sobre categorias para mostrar as categorias e os valores em cada uma delas.

```
fixName();
fixPrice();
checkQuantity();
var toFixJSON = JSON.stringify(bdBroken,null, " ");
saveJSON(toFixJSON);
```

E por último, mas não menos importante, chamamos as funções 'fixName', 'fixPrice', 'checkQuantity', que rodaram os consertos do JSON.

Na variável 'toFixJSON', passamos o método 'JSON.stringify' para que o 'bdBroken' seja passado como um objeto padrão para o método 'fs.writeFile' e não tenha erros.

```
function saveJSON(jsontoString){
  fs.writeFile("saida.json", jsontoString, function(err) {
    if (err) {
      console.log(err);
    }else{
      sortJSON();
      quotePrice();
    }
  });
}
```

Você deve estar se perguntando, mas por que a chamada das funções 'sortJSON()' e 'quotePrice()', foram feitas dentro da função callback do método 'fs.writeFile'?

Pois, acontece que se chamarmos essas funções em segunda da 'saveJSON', que é a função que cria o arquivo com o método 'fs.writeFile', ele vai dar erro, pois a chamada das funções será tão rápida que não dará tempo de criar o novo arquivo JSON, e assim dando erro.

Passando dentro da função callback do método 'fs.writeFile', primeiro ele cria o arquivo, depois executa a função callback.

OBS:

O que mais tive dificuldade foi trabalhar com os métodos 'map()', 'filter()' e 'reduce()', por mais que haja bons tutoriais na internet, devido a ser um caso bem específico ficou bem difícil de abstrair como seria feito.

Foi escolhido JavaScript, pois com o mercado tendo bastante demanda nessa linguagem, a gama de conteúdos e tutoriais é maior.

Desde já, agradeço pela oportunidade.