Vaga: Estágio - TI: Web Development

Candidato: Caio Teruo Fuzita Rabelo da Silva

1. Criação das funções

1.1. Módulo *file system*

```
let fs = require('fs'); // inclui o modulo 'file system'
```

Módulo necessário para abrir e salvar arquivos locais.

1.2. Abrir o arquivo .json

```
47  // lê o arquivo .json
48  function readFileJson() {
49      try{
50          const data = fs.readFileSync('./broken-database.json','utf8');
51          return JSON.parse(data); // fonte: https://developer.mozilla.or
52      } catch (err){
53          console.error(err)
54      }
55  }
```

O método .readFileSync é o responsável por abrir o arquivo ‘broken-database.json’, já o método .parse lê o arquivo .json e retorna como um objeto javascript. Caso não seja possível abrir o arquivo, será exibido uma mensagem de erro, capturado pelo try e o catch.

1.3. Corrigir as letras

```
57  // corrige as letras
58  function fixName(letters) {
59      return letters.replace(/æ/g,'a').replace(/¢/g,'c').replace(/ø/g,'o').replace(/ß/g,'b');
60  }
```

O método .replace troca os caracteres do primeiro termo pelo segundo, e o ‘g’ faz com que a mudança seja global, em todas as ocorrências presentes no arquivo.

1.4. Corrigir os preços

```
// corrige os preços
function fixPrice(preco){
    return Number(preco);
}
```

A função *Number()* realiza a conversão de um argumento para um número, nesse caso, converte os argumentos de ‘preco’ para o tipo Number.

1.5. Corrigir as quantidades

```
// corrige as quantidades
function fixQuantity(quantity){
    return quantity || 0;
}
```

Essa função analisa o argumento ‘quantity’, se um valor aparecer primeiro, no caso de quantity ≠ 0, esse valor será atribuído, caso contrário, é inserido o valor 0.

1.6. Salvar o arquivo .json

```
// salva o arquivo em .json
function saveFileJson(file){
    let dados = JSON.stringify(file);
    fs.writeFileSync('./saida.json', dados);
}
```

O método .stringify converte valores em javascript para uma string .json. O método .writeFileSync é semelhante ao método presente na função para abrir o arquivo .json. Faz com que o arquivo salvo tenha o nome de ‘saida.json’.

1.7. Abrir o arquivo .json corrigido

```
78  // abre o arquivo em .json já corrigido
79  function readFixedJson() {
80      try{
81          const data = fs.readFileSync('./saida.json','utf8');
82          return JSON.parse(data);
83      } catch (err){
84          console.error(err)
85      }
86  }
```

Semelhante a função que abre o arquivo .json quebrado, apenas mudando o arquivo que será aberto.

1.8. Calcular o valor do estoque

```
88  // faz a multiplicacao (quantidade x preço) de um objeto
89  function calculaEstoque(products){
90      let soma = 0;
91      for (let i in products){
92          soma += products[i]['quantity'] * products[i]['price'];
93      } return soma
94  }
```

Realiza um loop, o valor inicial da soma é 0, sendo acrescido por cada repetição presente no argumento ‘products’, cada array que aparece tem sua quantidade multiplicada pelo preço e acrescido na variável soma.

2. Execução

2.1. Abrindo o arquivo

```
3   // abrindo o arquivo .json
4   let brokenJson = readFileJson();
```

Criação da variável chamada ‘brokenJson’ para receber o arquivo .json.

2.2. Aplicando as correções

```
6   //realizando as devidas correções no arquivo
7   for (let i in brokenJson){
8       brokenJson[i]['name'] = fixName(brokenJson[i]['name']);
9       brokenJson[i]['price'] = fixPrice(brokenJson[i]['price']);
10      brokenJson[i]['quantity'] = fixQuantity(brokenJson[i]['quantity']);
11  }
```

Realiza uma varredura sobre o arquivo com a função for, cada função funcionará em cima de sua array correspondente.

2.3. Salvando o arquivo corrigido

```
13  // salvando o arquivo de forma corrigida
14  saveFileJson(brokenJson);
```

```
16  // abrindo o arquivo corrigido
17  let fixedJson = readFixedJson();
```

Utiliza a função para salvar o arquivo já corrigido com o nome de ‘saida.json’, em seguida, o arquivo corrigido é retornado para continuar a validação.

2.4. Classificando por ID e por categoria

```
19  // ordenando por categoria e id
20  fixedJson.sort(function (a, b){
21      return (a.category > b.category) ? 1 : ((b.category > a.category) ? -1 : (a.id > b.id) ? 1 : (b.id > a.id) ? -1 : 0);
22  });
```

```
24  // imprimindo o arquivo ordenado
25  console.log('Ordenado por categoria e ID:')
26  console.log('')
27  console.log(fixedJson);
```

Uso do método sort. Ocorre primeiramente a classificação por ordem alfabética (A-Z) e, logo após, por ID em ordem crescente. Em seguida o objeto é impresso no console conforme a validação.

2.5. Listando as categorias existentes

```
33  // listando as categorias existentes
34  const categorias = ['Acessórios','Eletrodomésticos','Eletrônicos','Panelas'];
```

É criada uma constante com as categorias existentes no banco de dados para o auxílio no cálculo de valor do estoque.

2.6. Varredura e cálculo do valor de estoque por categoria

```
36  // varrendo e aplicando a função para calcular o valor do estoque de cada categoria
37  for (let i = 0; i < categorias.length; i++) {
38      let produtos = fixedJson.filter((search) => {
39          return search.category == categorias[i];
40      })
41      console.log(categorias[i], calculaEstoque(produtos))
42  }
```

Um loop for é criado para cada índice da array categorias. O método filter é responsável por filtrar o arquivo corrigido e trazer somente os objetos nos quais o categoria[i] está presente, chamando a função de calcular estoque, e imprimindo o valor do estoque inteiro separado por categoria.

3. JavaScript

A escolha da linguagem JavaScript deve-se ao fato dessa linguagem ser otimizada para manipulação, cálculo e validação de informações que são enviadas e recebidas, principalmente com o formato .json muito presente nos bancos de dados. Além da alta compatibilidade existente, pois os sistemas funcionam em praticamente todos os navegadores atuais, também é utilizada para o desenvolvimento web, área necessária para praticamente todas as empresas atualmente.