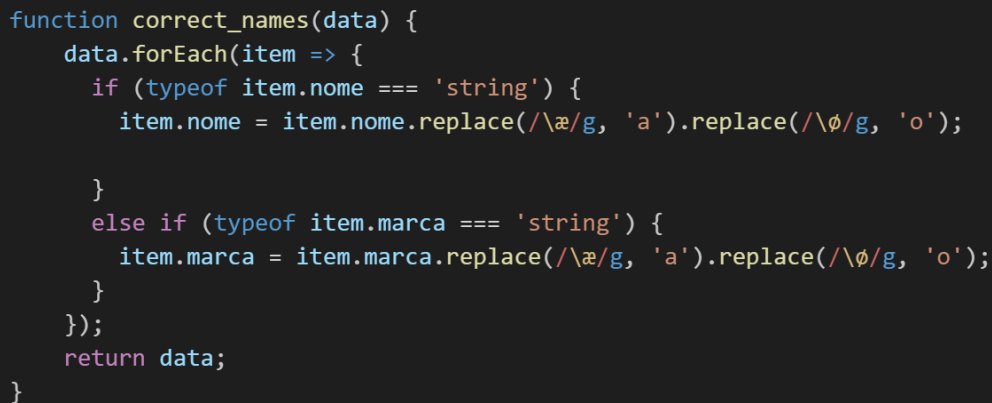


Documento de análise e explicação de código

1. Função para corrigir nome de marcas e veículos

1.1 Código da Função:



```
function correct_names(data) {  
  data.forEach(item => {  
    if (typeof item.nome === 'string') {  
      item.nome = item.nome.replace(/\æ/g, 'a').replace(/\ø/g, 'o');  
    }  
    else if (typeof item.marca === 'string') {  
      item.marca = item.marca.replace(/\æ/g, 'a').replace(/\ø/g, 'o');  
    }  
  });  
  return data;  
}
```

1.2 Explicação da Função

A função **correct_names** é responsável por corrigir caracteres específicos em strings associadas a nomes de marcas e veículos dentro de um array de objetos.

- **Objetivo:** Corrigir caracteres especiais (æ e ø) substituindo-os por a e o, respectivamente.
- **Parâmetro:** **data** - um array de objetos onde cada objeto pode ter propriedades **nome** e **marca**.
- **Lógica:**
 1. Itera sobre cada item do array com o método **forEach**.
 2. Se a propriedade **nome** é uma string, substitui æ por a e ø por o.
 3. Se a propriedade **marca** é uma string, faz as mesmas substituições.

1.3 Tratamentos de Bugs

- **Verificação de Tipo:** O código verifica se `item.nome` ou `item.marca` é uma string antes de aplicar as substituições. Isso evita erros ao tentar usar `replace` em tipos de dados diferentes de strings.
- **Evitando Substituições Indesejadas:** As substituições são feitas apenas em `nome` e `marca`, com base no tipo de dado, garantindo que outras propriedades não sejam alteradas inadvertidamente.

2. Função para Corrigir Vendas

2.1 Código da Função:

```
function correct_sales(data){  
  data.forEach(item => {  
    if (typeof item.vendas === 'string') {  
      item.vendas = Number(item.vendas);  
    }  
  });  
  return data;  
}
```

2.2 Explicação da Função

A função `correct_sales` é responsável por converter valores de vendas que estão como strings em números.

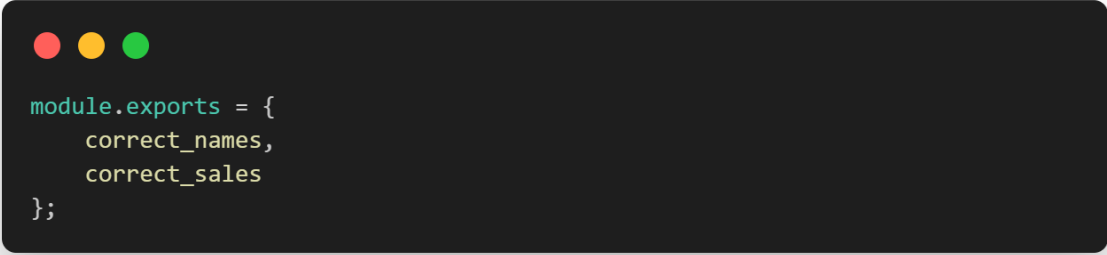
- **Objetivo:** Garantir que os valores da propriedade `vendas` sejam do tipo numérico para permitir operações matemáticas.
- **Parâmetro:** `data` - um array de objetos onde cada objeto pode ter a propriedade `vendas`.
- **Lógica:**
 1. Itera sobre cada item do array com o método `forEach`.
 2. Se a propriedade `vendas` é uma string, converte o valor para um número usando `Number()`.

2.3 Tratamentos de Bugs

- **Conversão de Tipo:** O código verifica se `item.vendas` é uma string antes de converter para número, evitando a conversão desnecessária de dados que já estão no formato numérico.
- **Não Manipula Outras Propriedades:** A função só altera a propriedade `vendas`, garantindo que outras propriedades dos objetos no array não sejam afetadas.

3. Exportando as Funções

3.1 Código de Exportação

A dark-themed code editor window with three colored window control buttons (red, yellow, green) in the top-left corner. It contains a few lines of JavaScript code for exporting functions.

```
module.exports = {  
  correct_names,  
  correct_sales  
};
```

3.2 Explicação da Exportação

A exportação é realizada com `module.exports`, permitindo que as funções `correct_names` e `correct_sales` sejam utilizadas em outros módulos ou arquivos JavaScript.

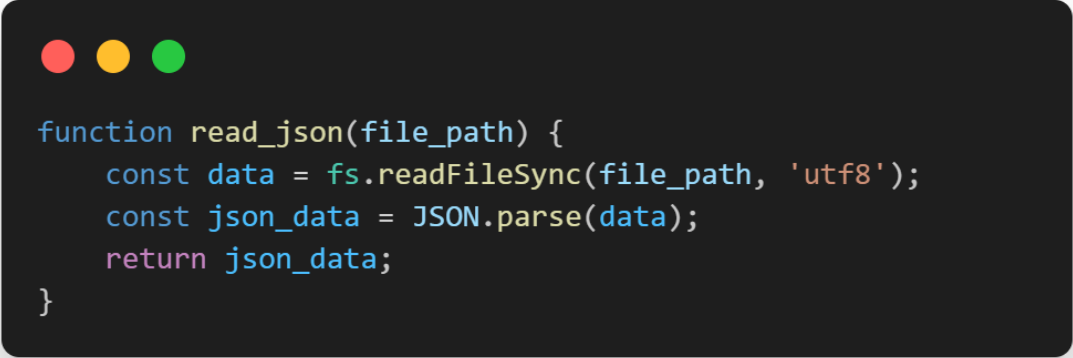
- **Objetivo:** Tornar as funções disponíveis para importação em outros scripts.
- **Lógica:** Define um objeto com as funções como propriedades e exporta esse objeto.

3.3 Tratamentos de Bugs

- **Exportação Correta:** A exportação é feita de maneira a garantir que ambas as funções possam ser acessadas e usadas em diferentes partes do código.

4. Função para Ler Arquivos JSON

4.1 Código da Função:



```
function read_json(file_path) {  
  const data = fs.readFileSync(file_path, 'utf8');  
  const json_data = JSON.parse(data);  
  return json_data;  
}
```

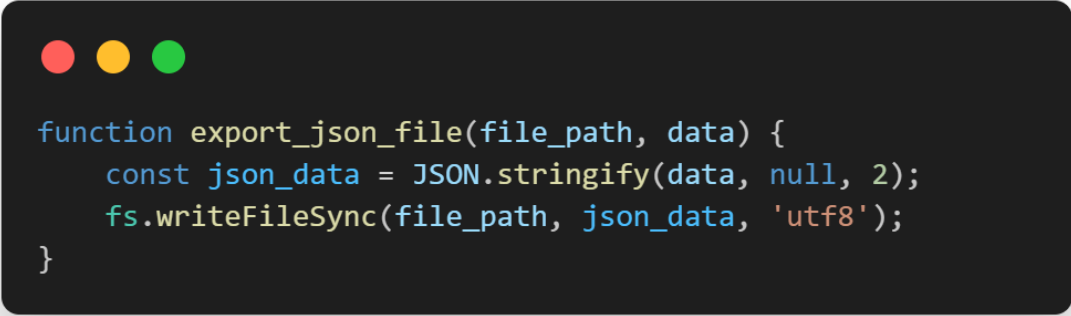
4.2 Explicação da Função

A função **read_json** lê um arquivo JSON e o converte para um objeto JavaScript.

- **Objetivo:** Ler o conteúdo de um arquivo JSON e convertê-lo para um formato de dados utilizável.
- **Parâmetro:** **file_path** - O caminho para o arquivo JSON que deve ser lido.
- **Lógica:**
 1. Usa **fs.readFileSync** para ler o conteúdo do arquivo de forma síncrona.
 2. Converte o conteúdo do arquivo de JSON para um objeto JavaScript com **JSON.parse**.
 3. Retorna o objeto JavaScript.

5. Função para Escrever Arquivos JSON

5.1 Código da Função:



```
function export_json_file(file_path, data) {  
    const json_data = JSON.stringify(data, null, 2);  
    fs.writeFileSync(file_path, json_data, 'utf8');  
}
```

5.2 Explicação da Função

A função **export_json_file** escreve um objeto JavaScript em um arquivo JSON.

- **Objetivo:** Converter um objeto JavaScript para JSON e gravá-lo em um arquivo.
- **Parâmetro:**
 1. **file_path:** O caminho para o arquivo JSON onde os dados serão salvos.
 2. **data:** O objeto JavaScript que deve ser convertido para JSON e escrito no arquivo.
- **Lógica:**
 1. Converte o objeto JavaScript para uma string JSON formatada com **JSON.stringify**.
 2. Grava a string JSON no arquivo especificado com **fs.writeFileSync**.

6. Função para Gerar SQL INSERT

6.1 Código da Função:

```
function generate_sql_insert(data1, data2, flag) {  
  return data1.map(item1 => {  
    let { data, id_marca_, nome, valor_do_veiculo,  
    vendas, marca, id_marca } = item1;  
  
    const datasql = get_sql_value(data);  
    const id_modelosql = get_sql_value(id_marca_);  
    const nome_modelosql = get_sql_value(nome);  
    const valor_do_veiculosql = get_sql_value(  
valor_do_veiculo);  
    const vendassql = get_sql_value(vendas);  
  
    if (flag == 3) {  
      marca = find_brand(data2, id_marca_);  
      id_marca = get_sql_value(id_marca_);  
    }  
  
    const marca_sql = get_sql_value(marca);  
    const id_marca_sql = get_sql_value(id_marca);  
    const total_de_vendassql = get_sql_value(  
valor_do_veiculo * vendas);
```

```

        if (flag === 1) {
            return
            `INSERT INTO Dados_de_vendas (data_da_venda, id_modelo,
            nome_modelo, valor_do_veiculo, vendas) VALUES (
            ${datasql}, ${id_modelosql}, ${nome_modelosql}, ${
            valor_do_veiculosql}, ${vendassql});`;
        } else if (flag === 2) {
            return
            `INSERT INTO Dados_de_vendas (id_marca, marca) VALUES (
            ${id_marca_sql}, ${marca_sql});`;
        } else if (flag === 3) {
            return
            `INSERT INTO Dados_de_vendas (data_da_venda, id_marca, m
            arca, nome_modelo, valor_do_veiculo, vendas, total_de_ve
            ndas) VALUES (
            ${datasql}, ${id_marca_sql}, ${marca_sql}, ${
            nome_modelosql}, ${valor_do_veiculosql}, ${vendassql},
            ${total_de_vendassql});`;
        } else {
            return null;
        }
    }).filter(sql => sql !== null).join('\n');
}

```

6.2 Explicação da Função

A função **generate_sql_insert** cria instruções SQL para inserir dados em uma tabela de banco de dados com base em um array de dados e um parâmetro de controle (**flag**).

- **Objetivo:** Gerar instruções SQL **INSERT** para adicionar dados à tabela **Dados_de_vendas**.
- **Parâmetros:**
 1. **data1:** Array de objetos com informações de vendas e modelos.
 2. **data2:** Array de objetos com informações de marcas.
 3. **flag:** Valor que determina o tipo de **INSERT** a ser gerado.
- **Lógica:**
 1. Itera sobre **data1** para gerar instruções **INSERT** dependendo do valor de **flag**.


2. Usa `get_sql_value` para tratar valores dos dados e prevenir SQL injection.
3. Se `flag` é 3, também atualiza `marca` e `id_marca` com dados de `data2`.

6.3 Tratamentos de Bugs

- **Validação de Dados:** `get_sql_value` e `is_empty` ajudam a evitar erros na construção das queries SQL e prevenirem SQL injection.

7. Exportando as Funções

7.1 Código de Exportação



```
module.exports = {  
  read_json,  
  export_json_file,  
  generate_sql_insert  
};
```


7.2 Explicação da Exportação

A exportação é realizada para que as funções possam ser usadas em outros arquivos ou módulos.

- **Objetivo:** Tornar as funções `read_json`, `export_json_file` e `generate_sql_insert` disponíveis para outros módulos.
- **Lógica:** Define um objeto com as funções e exporta esse objeto para que possa ser importado onde necessário.

8. Código do Fluxo Principal

8.1 Importação das Funções:




```
const { read_json, export_json_file, generate_sql_insert
  } = require('./file_operations');
const { correct_names, correct_sales } = require(
  './data_correction');
```

Objetivo: Importar funções dos módulos `file_operations` e `data_correction` para usar em nosso fluxo principal.

Funções Importadas:

- `read_json`: Lê arquivos JSON.
- `export_json_file`: Escreve arquivos JSON.
- `generate_sql_insert`: Gera instruções SQL `INSERT`.
- `correct_names`: Corrige nomes de marcas e veículos.
- `correct_sales`: Corrige valores de vendas.

8.2 Leitura dos Arquivos JSON



```
const db1 = read_json('./data/broken_database_1.json');
const db2 = read_json('./data/broken_database_2.json');
```

Objetivo: Ler os arquivos JSON contendo dados das bases de dados corrompidas.

8.3 Correção dos Dados



```
const db1_corrected = correct_names(db1);  
const db2_corrected = correct_names(db2);  
  
correct_sales(db1_corrected, db2_corrected);
```

Objetivo: Corrigir os dados lidos dos arquivos JSON.

8.4 Exportação dos Dados Corrigidos



```
export_json_file('./output/db1_corrected.json',  
db1_corrected);  
export_json_file('./output/db2_corrected.json',  
db2_corrected);
```

Objetivo: Salvar os dados corrigidos em arquivos JSON novos.

8.5 Geração dos Arquivos SQL



```
const db1_sql = generate_sql_insert(db1_corrected,  
db2_corrected, 1);  
const db2_sql = generate_sql_insert(db2_corrected,  
db1_corrected, 2);  
const db3_sql = generate_sql_insert(db1_corrected,  
db2_corrected, 3);
```

Objetivo: Gerar os scripts SQL **INSERT** para popular as tabelas do banco de dados.

8.6 Escrita dos Arquivos SQL

```
const db1_sql = generate_sql_insert(db1_corrected,  
db2_corrected,1);  
const db2_sql = generate_sql_insert(db2_corrected,  
db1_corrected, 2);  
const db3_sql = generate_sql_insert(db1_corrected,  
db2_corrected, 3);
```

8.7 Mensagem de Sucesso

```
console.log('\x1b[33m' +  
'-----');  
console.log('\x1b[33m' +  
'---- Arquivos gerados com sucesso! ----');  
console.log('\x1b[033m' +  
'-----');
```

Objetivo: Exibir uma mensagem indicando que os arquivos foram gerados com sucesso.