UC: Projetos - FrontEnd

DIAS: 17 e 18/10/2024 – Parte 2

Assunto:

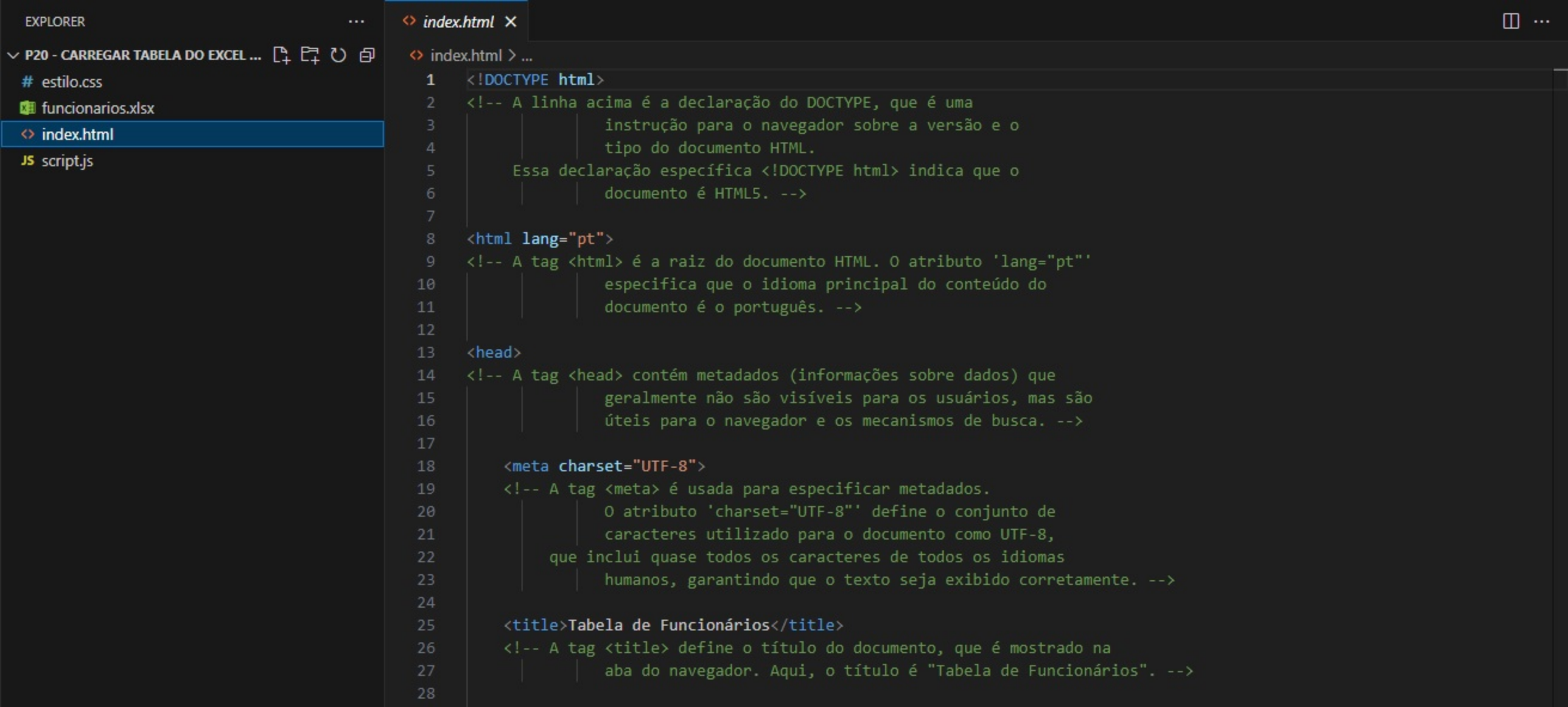
Projetos FrontEnd a Serem Desenvolvidos:

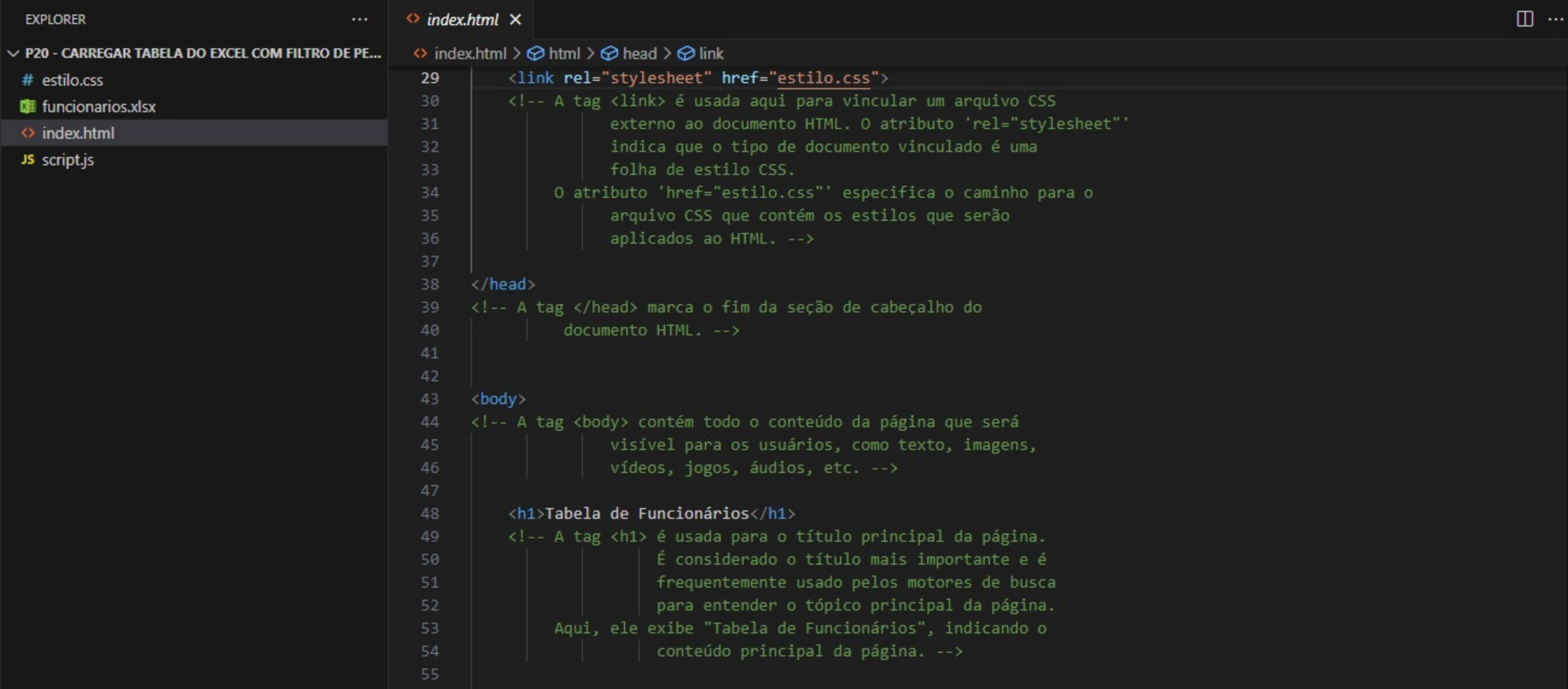
UC: Projetos - FrontEnd

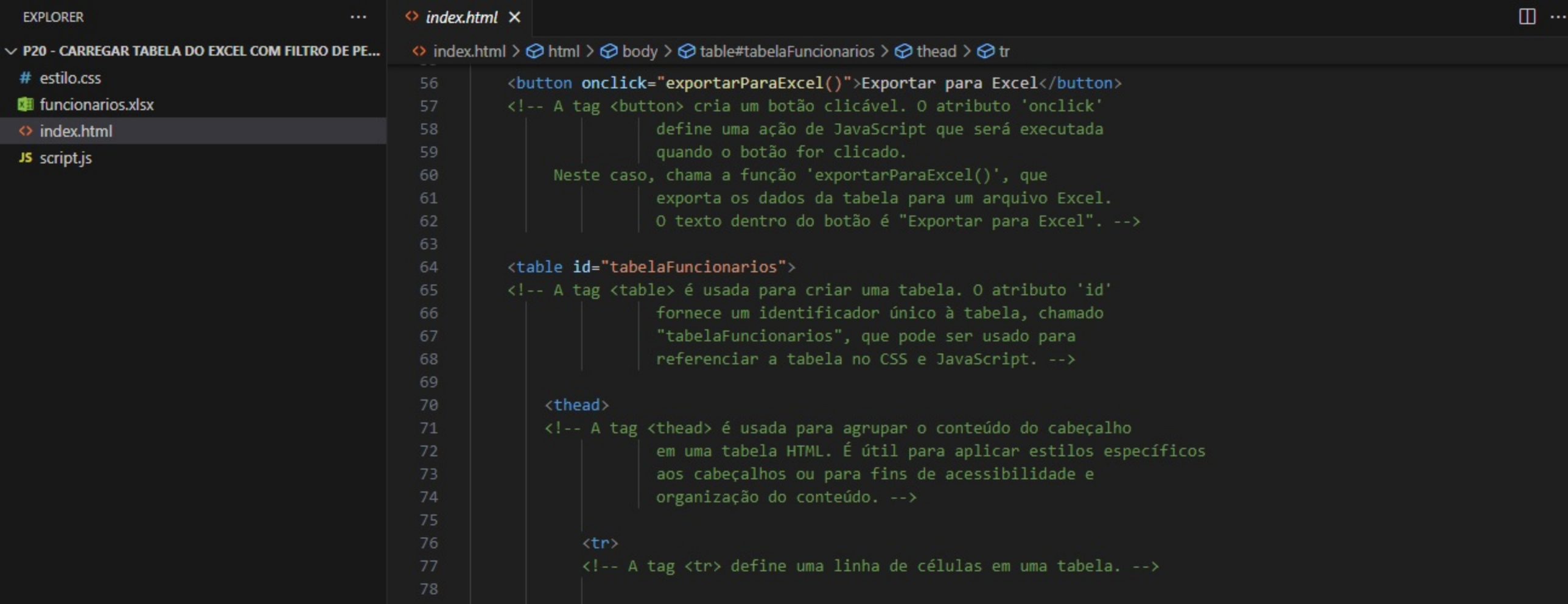
DIAS: 17 e 18/10/2024

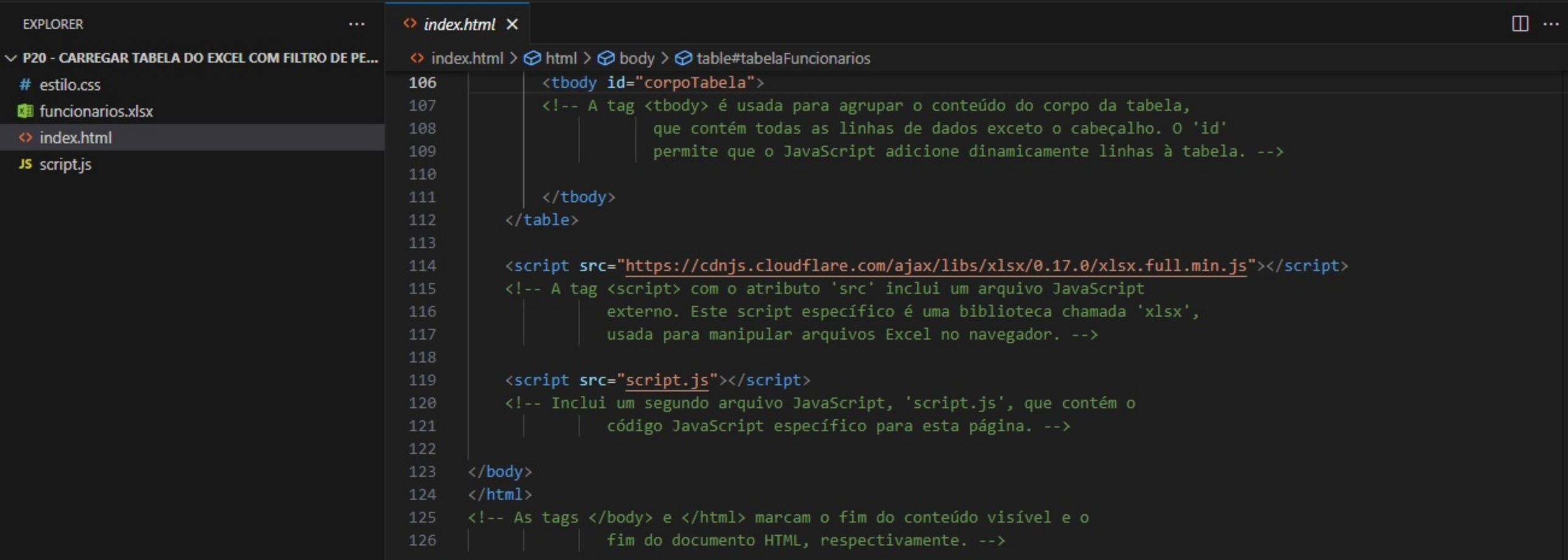
Assunto:

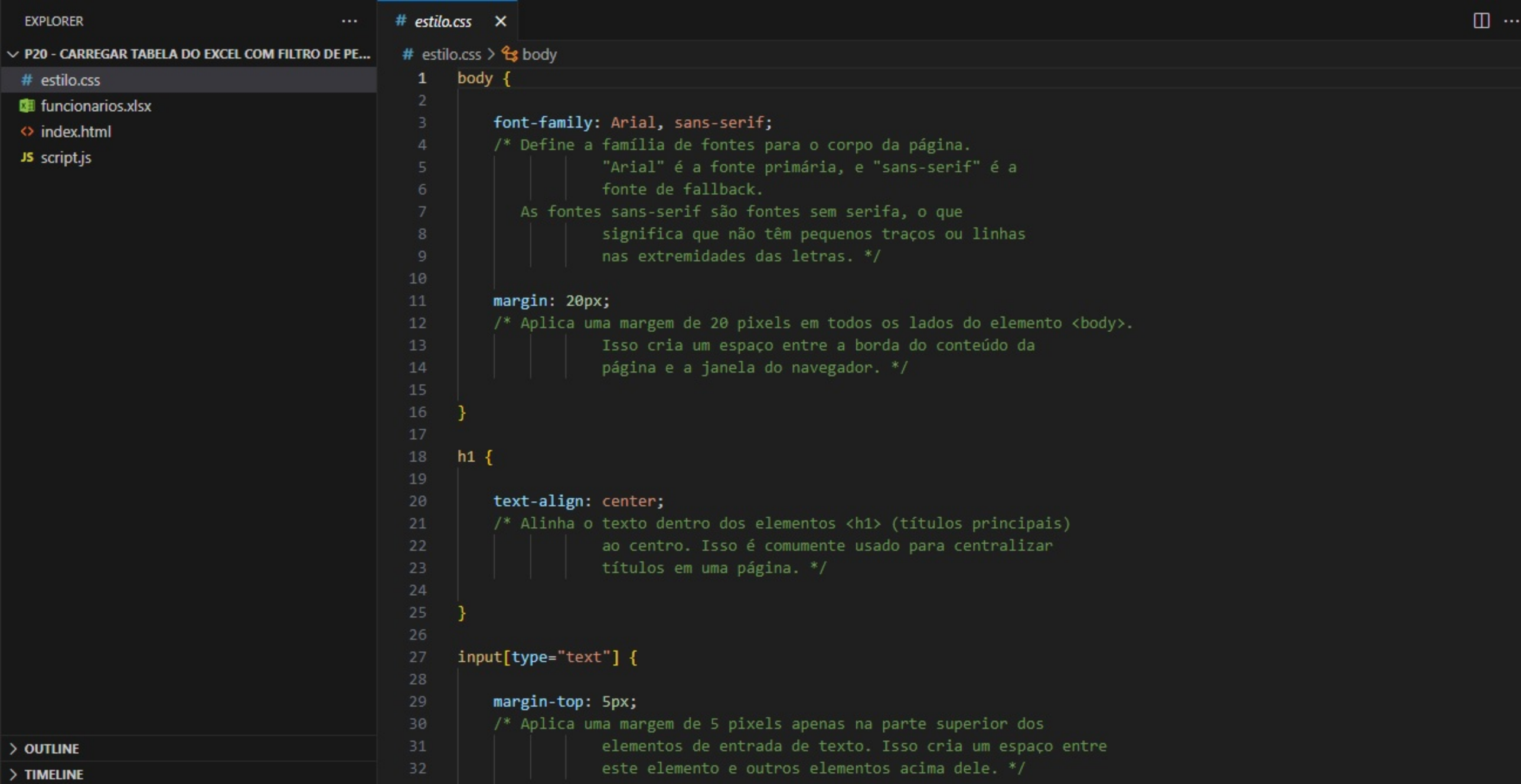
Projeto 20 – Carregar Tabela Excel com Filtro de Pesquisa

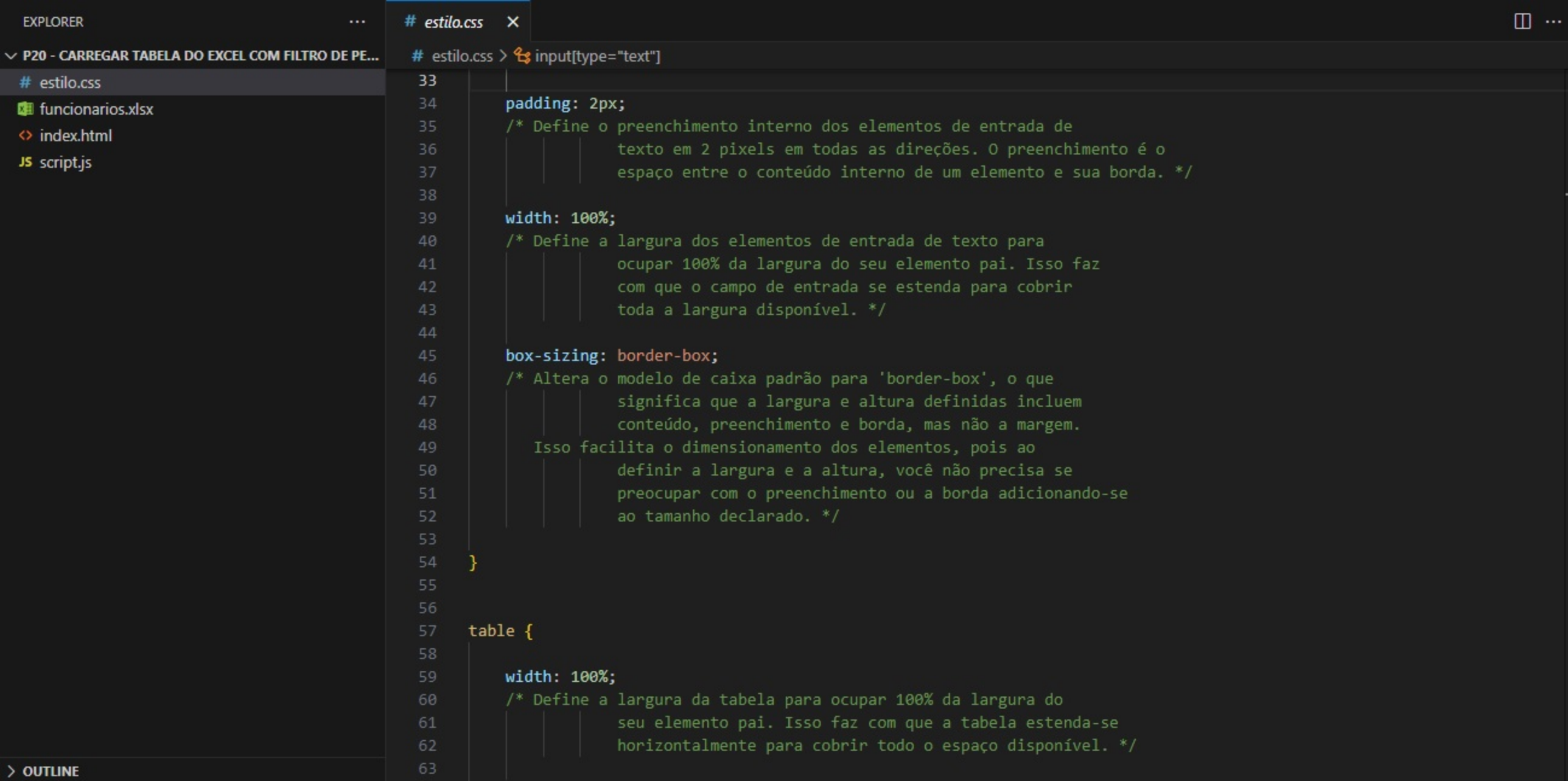

EXPLORER
P20 - CARREGAR TABELA DO EXCEL ...
estilo.css
funcionarios.xlsx
index.html
script.js
index.html
index.html > ...
1 <!DOCTYPE html>
2 <!-- A linha acima é a declaração do DOCTYPE, que é uma
3 instrução para o navegador sobre a versão e o
4 tipo do documento HTML.
5 Essa declaração específica <!DOCTYPE html> indica que o
6 documento é HTML5. -->
7
8 <html lang="pt">
9 <!-- A tag <html> é a raiz do documento HTML. O atributo 'lang="pt"'
10 especifica que o idioma principal do conteúdo do
11 documento é o português. -->
12
13 <head>
14 <!-- A tag <head> contém metadados (informações sobre dados) que
15 geralmente não são visíveis para os usuários, mas são
16 úteis para o navegador e os mecanismos de busca. -->
17
18 <meta charset="UTF-8">
19 <!-- A tag <meta> é usada para especificar metadados.
20 O atributo 'charset="UTF-8"' define o conjunto de
21 caracteres utilizado para o documento como UTF-8,
22 que inclui quase todos os caracteres de todos os idiomas
23 humanos, garantindo que o texto seja exibido corretamente. -->
24
25 <title>Tabela de Funcionários</title>
26 <!-- A tag <title> define o título do documento, que é mostrado na
27 aba do navegador. Aqui, o título é "Tabela de Funcionários". -->
28

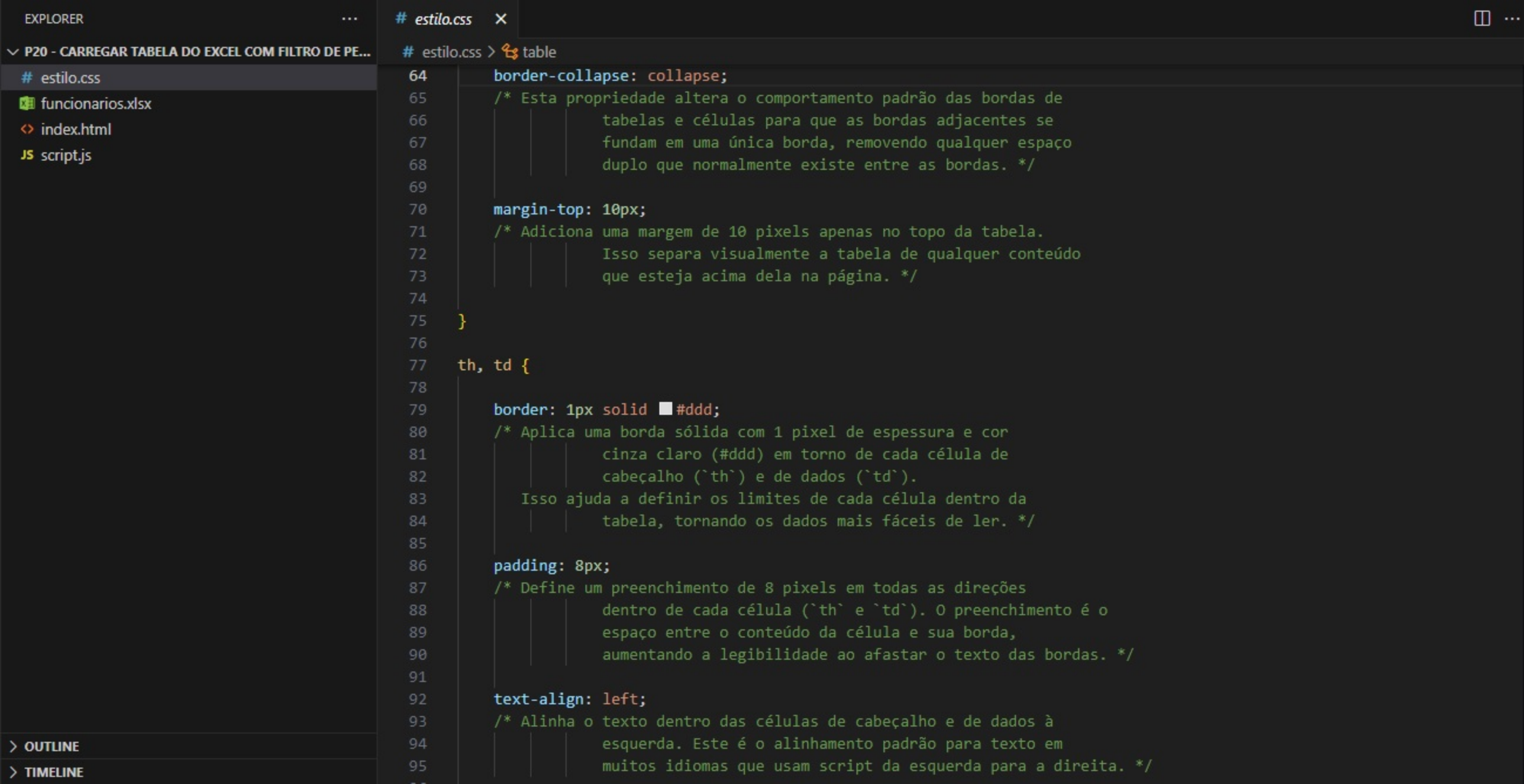

EXPLORER
index.html
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
index.html > html > head > link
<link rel="stylesheet" href="estilo.css">
<!-- A tag <link> é usada aqui para vincular um arquivo CSS externo ao documento HTML. O atributo 'rel="stylesheet"' indica que o tipo de documento vinculado é uma folha de estilo CSS. O atributo 'href="estilo.css"' especifica o caminho para o arquivo CSS que contém os estilos que serão aplicados ao HTML. -->
</head>
<!-- A tag </head> marca o fim da seção de cabeçalho do documento HTML. -->
<body>
<!-- A tag <body> contém todo o conteúdo da página que será visível para os usuários, como texto, imagens, vídeos, jogos, áudios, etc. -->
<h1>Tabela de Funcionários</h1>
<!-- A tag <h1> é usada para o título principal da página. É considerado o título mais importante e é frequentemente usado pelos motores de busca para entender o tópico principal da página. Aqui, ele exibe "Tabela de Funcionários", indicando o conteúdo principal da página. -->

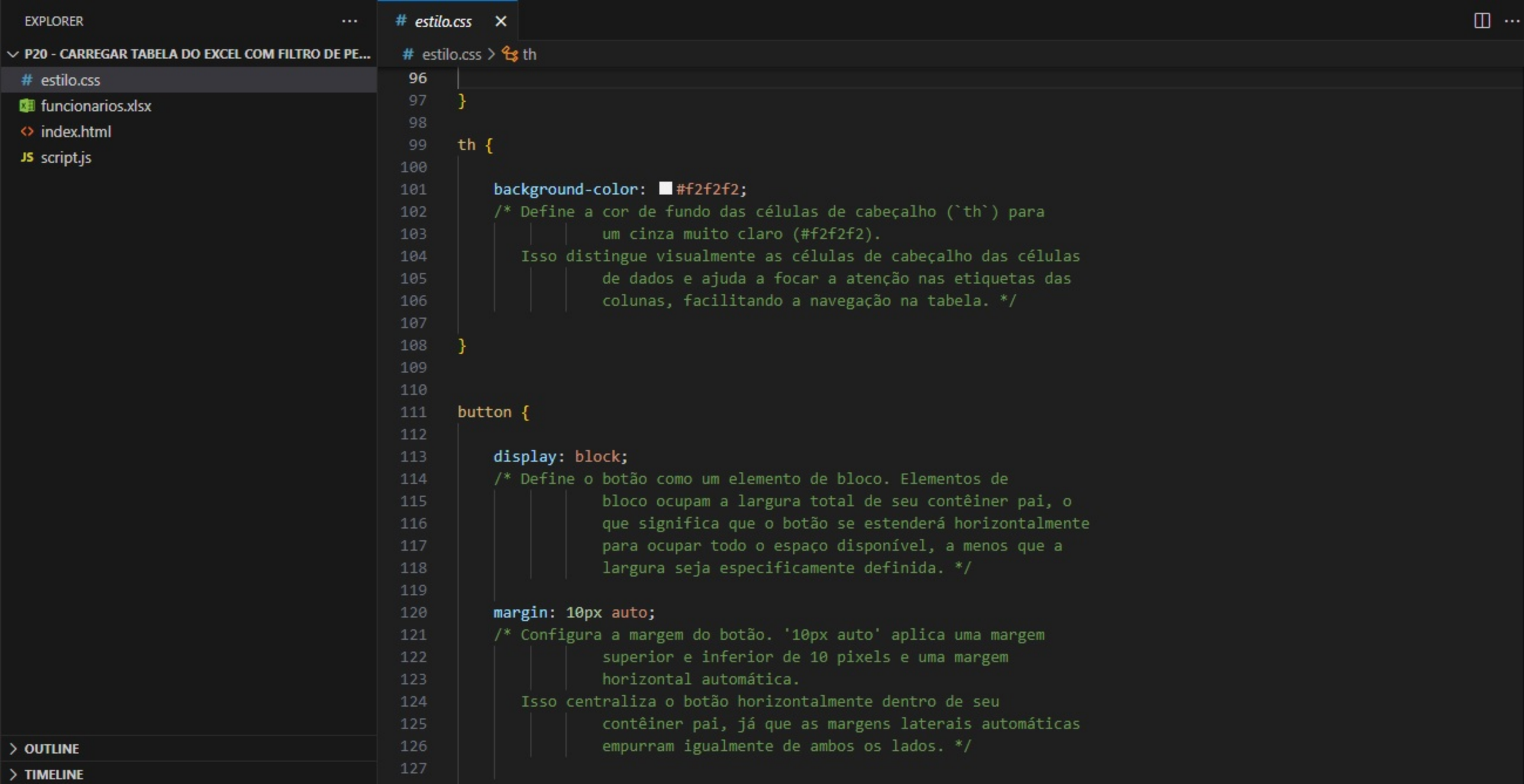

EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
index.html
index.html > html > body > table#tabelaFuncionarios > thead > tr
<button onclick="exportarParaExcel()">Exportar para Excel</button>
<!-- A tag <button> cria um botão clicável. O atributo 'onclick'
define uma ação de JavaScript que será executada
quando o botão for clicado.
Neste caso, chama a função 'exportarParaExcel()', que
exporta os dados da tabela para um arquivo Excel.
O texto dentro do botão é "Exportar para Excel". -->
<table id="tabelaFuncionarios">
<!-- A tag <table> é usada para criar uma tabela. O atributo 'id'
fornece um identificador único à tabela, chamado
"tabelaFuncionarios", que pode ser usado para
referenciar a tabela no CSS e JavaScript. -->
<thead>
<!-- A tag <thead> é usada para agrupar o conteúdo do cabeçalho
em uma tabela HTML. É útil para aplicar estilos específicos
aos cabeçalhos ou para fins de acessibilidade e
organização do conteúdo. -->
<tr>
<!-- A tag <tr> define uma linha de células em uma tabela. -->

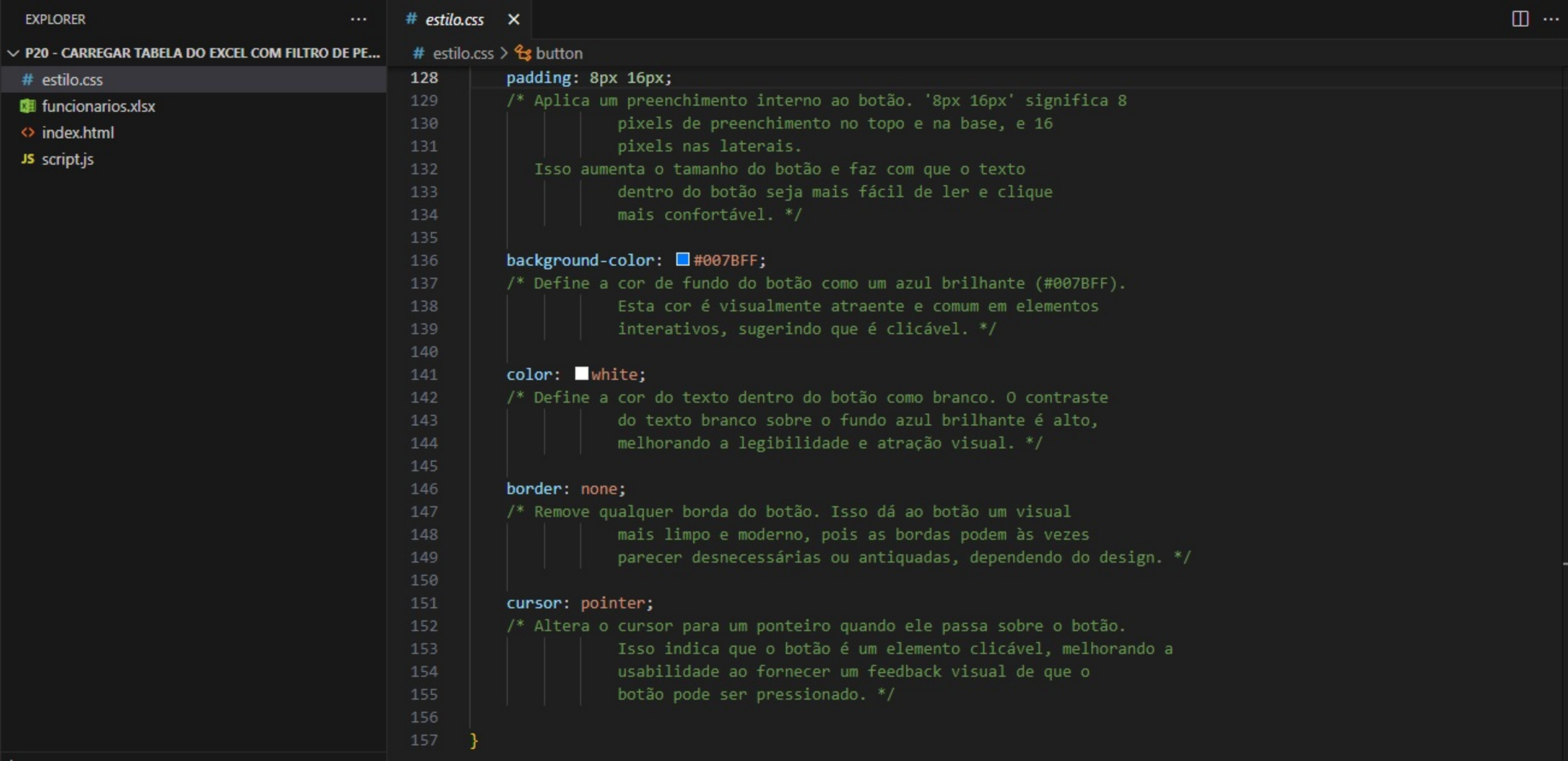

EXPLORER
index.html
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
index.html > html > body > table#tabelaFuncionarios
106 <tbody id="corpoTabela">
107 <!-- A tag <tbody> é usada para agrupar o conteúdo do corpo da tabela,
108 que contém todas as linhas de dados exceto o cabeçalho. O 'id'
109 permite que o JavaScript adicione dinamicamente linhas à tabela. -->
110
111 </tbody>
112 </table>
113
114 <script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.17.0/xlsx.full.min.js"></script>
115 <!-- A tag <script> com o atributo 'src' inclui um arquivo JavaScript
116 externo. Este script específico é uma biblioteca chamada 'xlsx',
117 usada para manipular arquivos Excel no navegador. -->
118
119 <script src="script.js"></script>
120 <!-- Inclui um segundo arquivo JavaScript, 'script.js', que contém o
121 código JavaScript específico para esta página. -->
122
123 </body>
124 </html>
125 <!-- As tags </body> e </html> marcam o fim do conteúdo visível e o
126 fim do documento HTML, respectivamente. -->

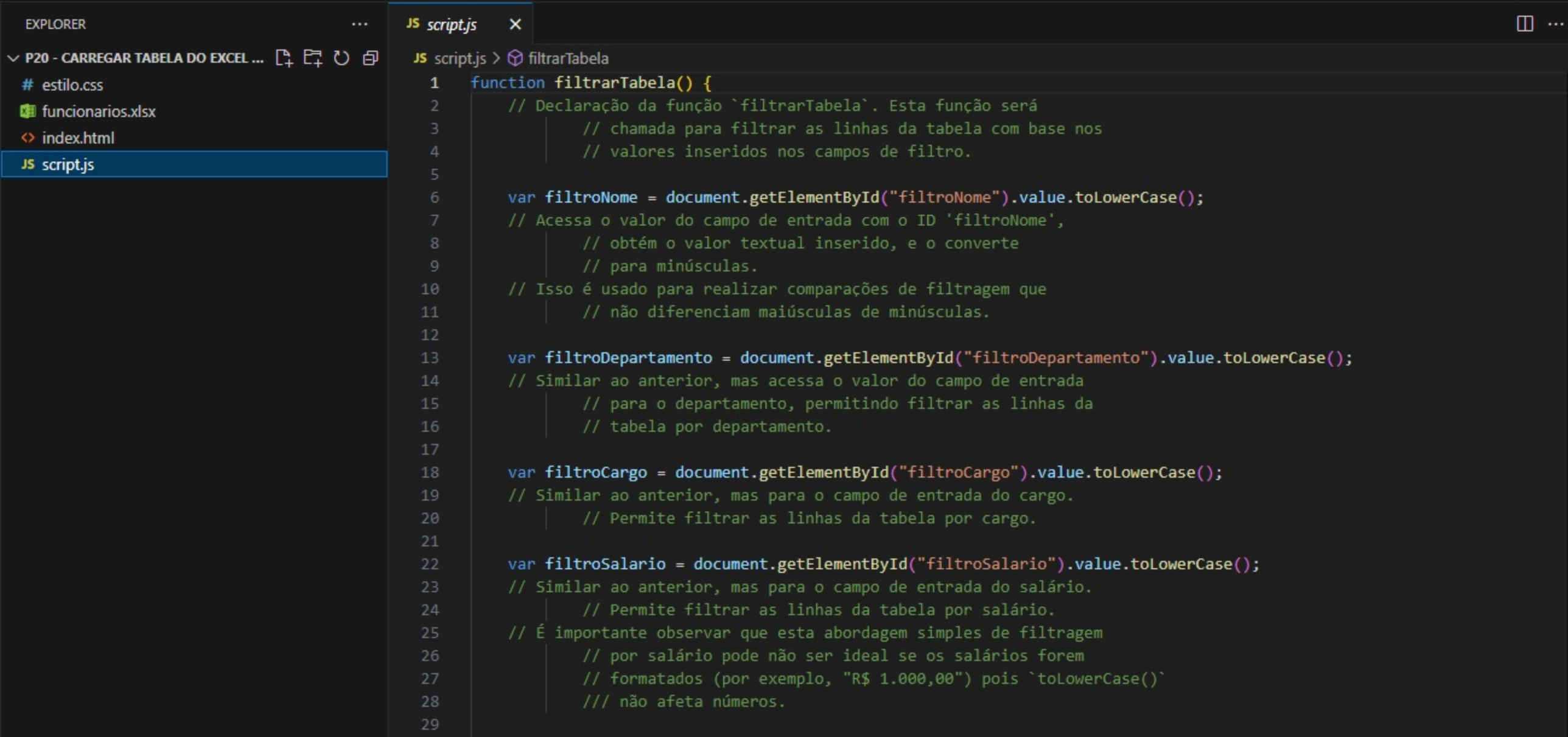

EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
OUTLINE
TIMELINE
estilo.css
estilo.css > body
body {
font-family: Arial, sans-serif;
/* Define a família de fontes para o corpo da página.
"Arial" é a fonte primária, e "sans-serif" é a
fonte de fallback.
As fontes sans-serif são fontes sem serifa, o que
significa que não têm pequenos traços ou linhas
nas extremidades das letras. */
margin: 20px;
/* Aplica uma margem de 20 pixels em todos os lados do elemento <body>.
Isso cria um espaço entre a borda do conteúdo da
página e a janela do navegador. */
}
h1 {
text-align: center;
/* Alinha o texto dentro dos elementos <h1> (títulos principais)
ao centro. Isso é comumente usado para centralizar
títulos em uma página. */
}
input[type="text"] {
margin-top: 5px;
/* Aplica uma margem de 5 pixels apenas na parte superior dos
elementos de entrada de texto. Isso cria um espaço entre
este elemento e outros elementos acima dele. */

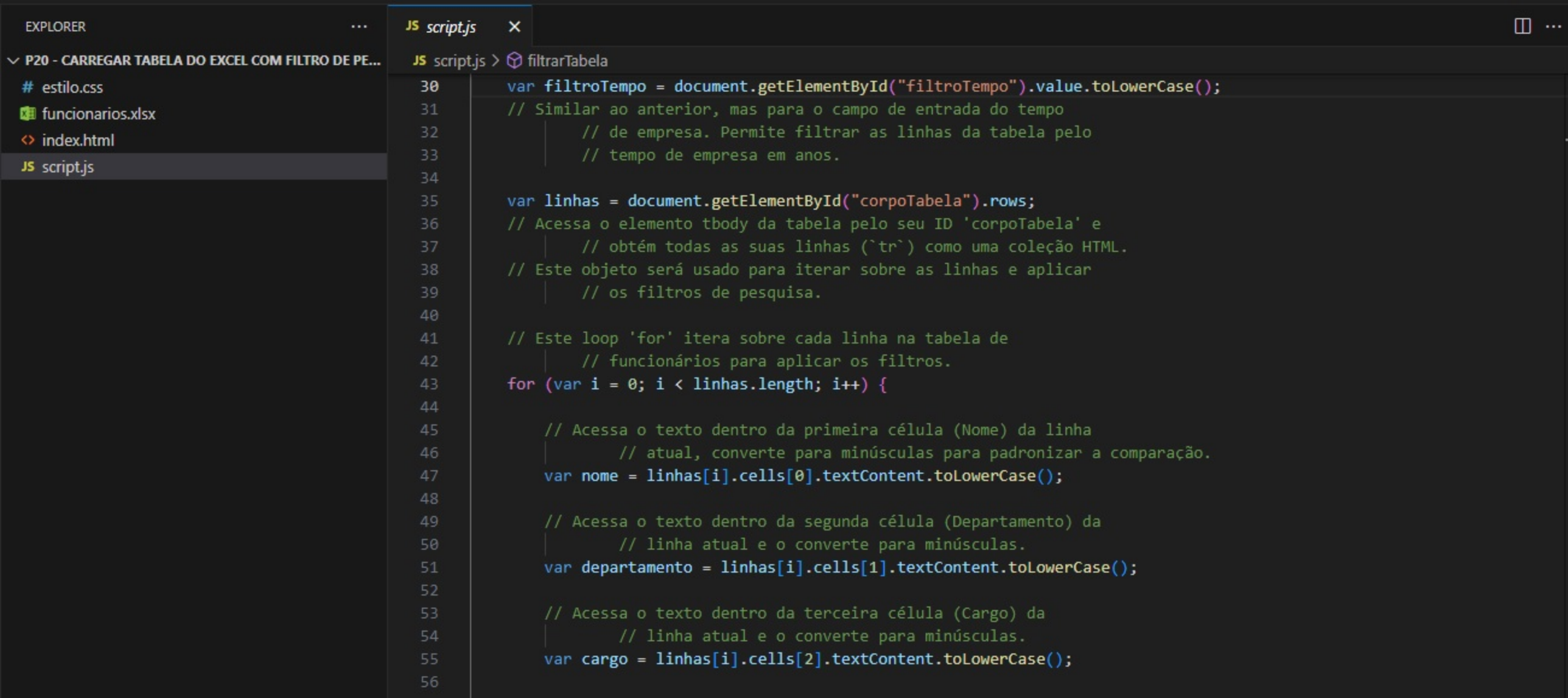

EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
OUTLINE
estilo.css
estilo.css > input[type="text"]
padding: 2px;
/* Define o preenchimento interno dos elementos de entrada de
texto em 2 pixels em todas as direções. O preenchimento é o
espaço entre o conteúdo interno de um elemento e sua borda. */
width: 100%;
/* Define a largura dos elementos de entrada de texto para
ocupar 100% da largura do seu elemento pai. Isso faz
com que o campo de entrada se estenda para cobrir
toda a largura disponível. */
box-sizing: border-box;
/* Altera o modelo de caixa padrão para 'border-box', o que
significa que a largura e altura definidas incluem
conteúdo, preenchimento e borda, mas não a margem.
Isso facilita o dimensionamento dos elementos, pois ao
definir a largura e a altura, você não precisa se
preocupar com o preenchimento ou a borda adicionando-se
ao tamanho declarado. */
}
table {
width: 100%;
/* Define a largura da tabela para ocupar 100% da largura do
seu elemento pai. Isso faz com que a tabela estenda-se
horizontalmente para cobrir todo o espaço disponível. */

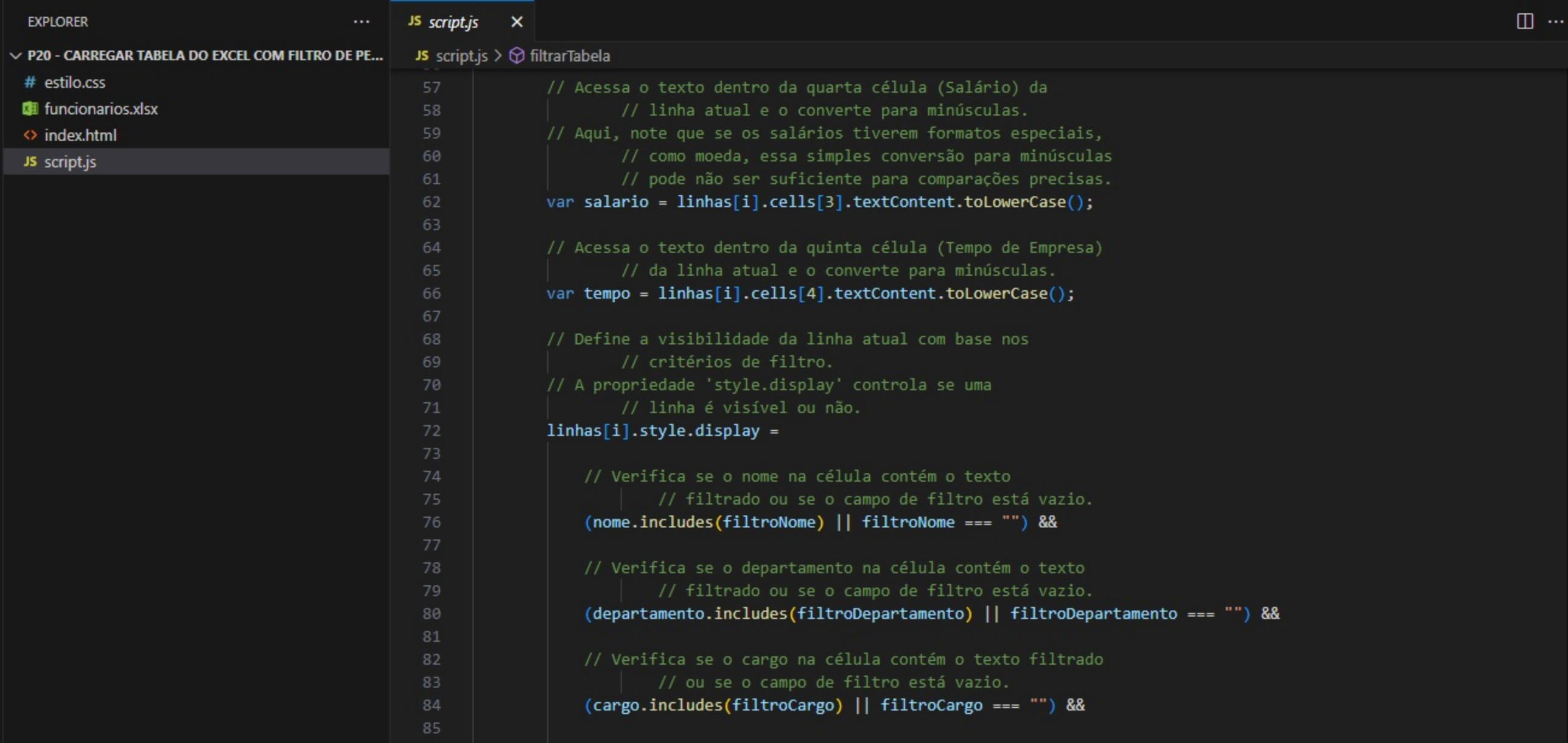

EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
OUTLINE
TIMELINE
estilo.css
estilo.css > table
border-collapse: collapse;
/* Esta propriedade altera o comportamento padrão das bordas de
tabelas e células para que as bordas adjacentes se
fundam em uma única borda, removendo qualquer espaço
duplo que normalmente existe entre as bordas. */
margin-top: 10px;
/* Adiciona uma margem de 10 pixels apenas no topo da tabela.
Isso separa visualmente a tabela de qualquer conteúdo
que esteja acima dela na página. */
}
th, td {
border: 1px solid #ddd;
/* Aplica uma borda sólida com 1 pixel de espessura e cor
cinza claro (#ddd) em torno de cada célula de
cabeçalho (`th`) e de dados (`td`).
Isso ajuda a definir os limites de cada célula dentro da
tabela, tornando os dados mais fáceis de ler. */
padding: 8px;
/* Define um preenchimento de 8 pixels em todas as direções
dentro de cada célula (`th` e `td`). O preenchimento é o
espaço entre o conteúdo da célula e sua borda,
aumentando a legibilidade ao afastar o texto das bordas. */
text-align: left;
/* Alinha o texto dentro das células de cabeçalho e de dados à
esquerda. Este é o alinhamento padrão para texto em
muitos idiomas que usam script da esquerda para a direita. */

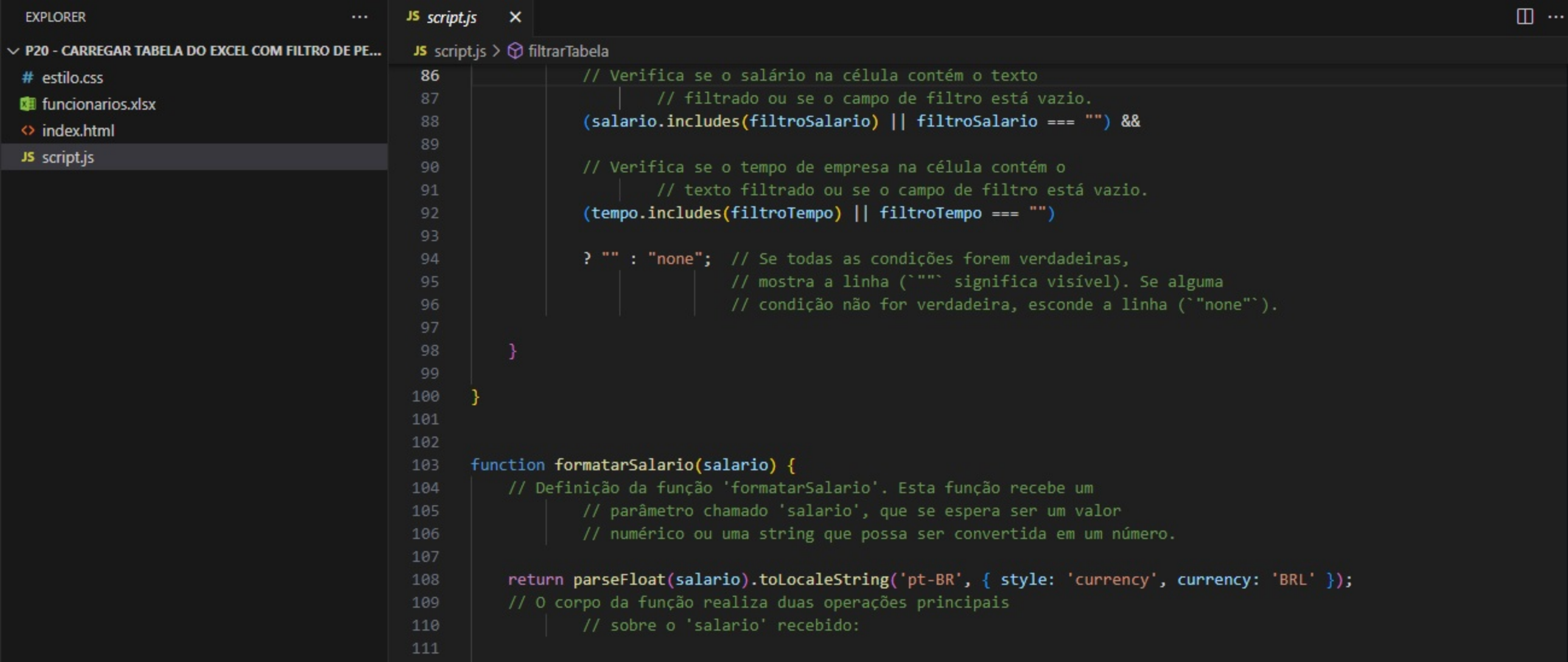

EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
OUTLINE
TIMELINE
estilo.css
estilo.css > th
}
th {
background-color: #f2f2f2;
/* Define a cor de fundo das células de cabeçalho (`th`) para
um cinza muito claro (#f2f2f2).
Isso distingue visualmente as células de cabeçalho das células
de dados e ajuda a focar a atenção nas etiquetas das
colunas, facilitando a navegação na tabela. */
}
button {
display: block;
/* Define o botão como um elemento de bloco. Elementos de
bloco ocupam a largura total de seu contêiner pai, o
que significa que o botão se estenderá horizontalmente
para ocupar todo o espaço disponível, a menos que a
largura seja especificamente definida. */
margin: 10px auto;
/* Configura a margem do botão. '10px auto' aplica uma margem
superior e inferior de 10 pixels e uma margem
horizontal automática.
Isso centraliza o botão horizontalmente dentro de seu
contêiner pai, já que as margens laterais automáticas
empurram igualmente de ambos os lados. */

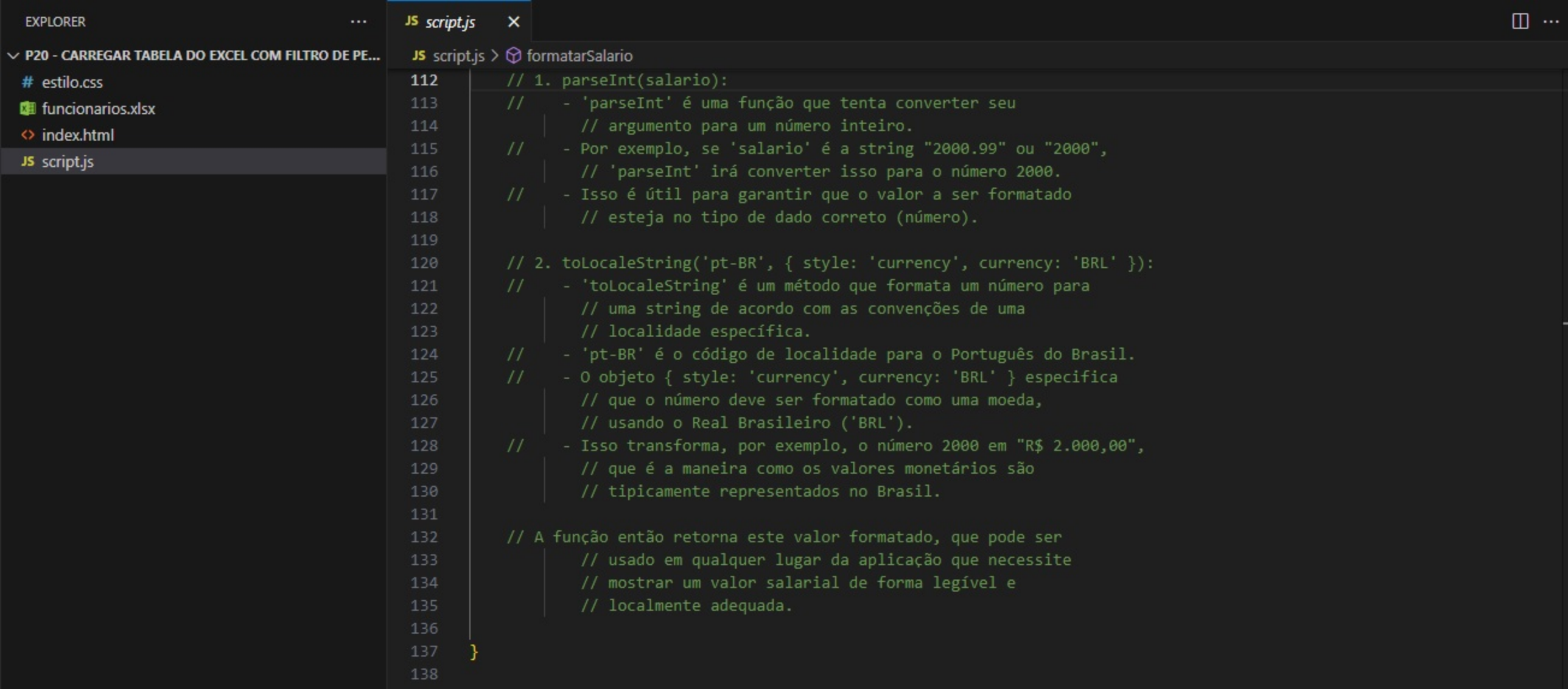

EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
estilo.css
estilo.css > button
padding: 8px 16px;
/* Aplica um preenchimento interno ao botão. '8px 16px' significa 8
pixels de preenchimento no topo e na base, e 16
pixels nas laterais.
Isso aumenta o tamanho do botão e faz com que o texto
dentro do botão seja mais fácil de ler e clique
mais confortável. */
background-color: #007BFF;
/* Define a cor de fundo do botão como um azul brilhante (#007BFF).
Esta cor é visualmente atraente e comum em elementos
interativos, sugerindo que é clicável. */
color: white;
/* Define a cor do texto dentro do botão como branco. O contraste
do texto branco sobre o fundo azul brilhante é alto,
melhorando a legibilidade e atração visual. */
border: none;
/* Remove qualquer borda do botão. Isso dá ao botão um visual
mais limpo e moderno, pois as bordas podem às vezes
parecer desnecessárias ou antiquadas, dependendo do design. */
cursor: pointer;
/* Altera o cursor para um ponteiro quando ele passa sobre o botão.
Isso indica que o botão é um elemento clicável, melhorando a
usabilidade ao fornecer um feedback visual de que o
botão pode ser pressionado. */
}

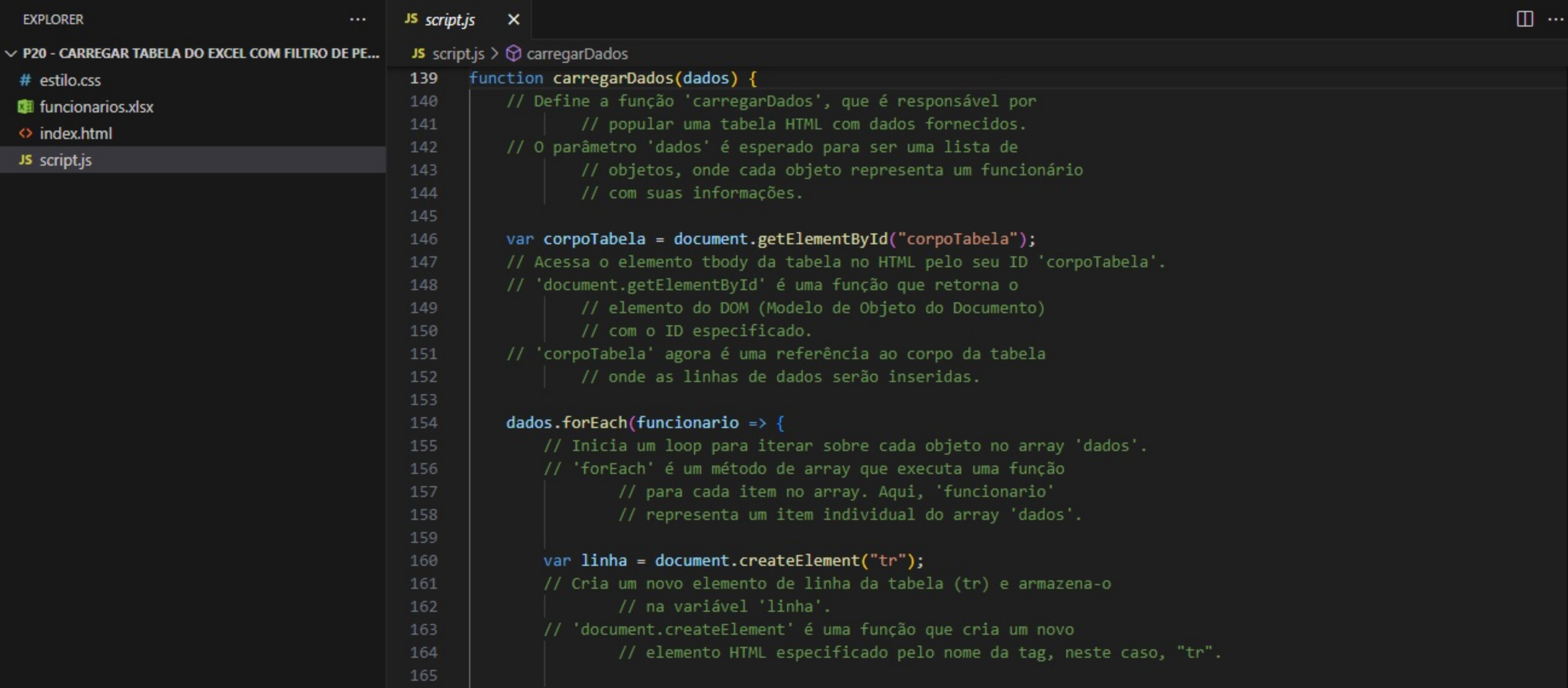

EXPLORER
P20 - CARREGAR TABELA DO EXCEL ...
estilo.css
funcionarios.xlsx
index.html
script.js
script.js
script.js > filtrarTabela
function filtrarTabela() {
// Declaração da função `filtrarTabela`. Esta função será
// chamada para filtrar as linhas da tabela com base nos
// valores inseridos nos campos de filtro.
var filtroNome = document.getElementById("filtroNome").value.toLowerCase();
// Acessa o valor do campo de entrada com o ID 'filtroNome',
// obtém o valor textual inserido, e o converte
// para minúsculas.
// Isso é usado para realizar comparações de filtragem que
// não diferenciam maiúsculas de minúsculas.
var filtroDepartamento = document.getElementById("filtroDepartamento").value.toLowerCase();
// Similar ao anterior, mas acessa o valor do campo de entrada
// para o departamento, permitindo filtrar as linhas da
// tabela por departamento.
var filtroCargo = document.getElementById("filtroCargo").value.toLowerCase();
// Similar ao anterior, mas para o campo de entrada do cargo.
// Permite filtrar as linhas da tabela por cargo.
var filtroSalario = document.getElementById("filtroSalario").value.toLowerCase();
// Similar ao anterior, mas para o campo de entrada do salário.
// Permite filtrar as linhas da tabela por salário.
// É importante observar que esta abordagem simples de filtragem
// por salário pode não ser ideal se os salários forem
// formatados (por exemplo, "R$ 1.000,00") pois `toLowerCase()`
/// não afeta números.

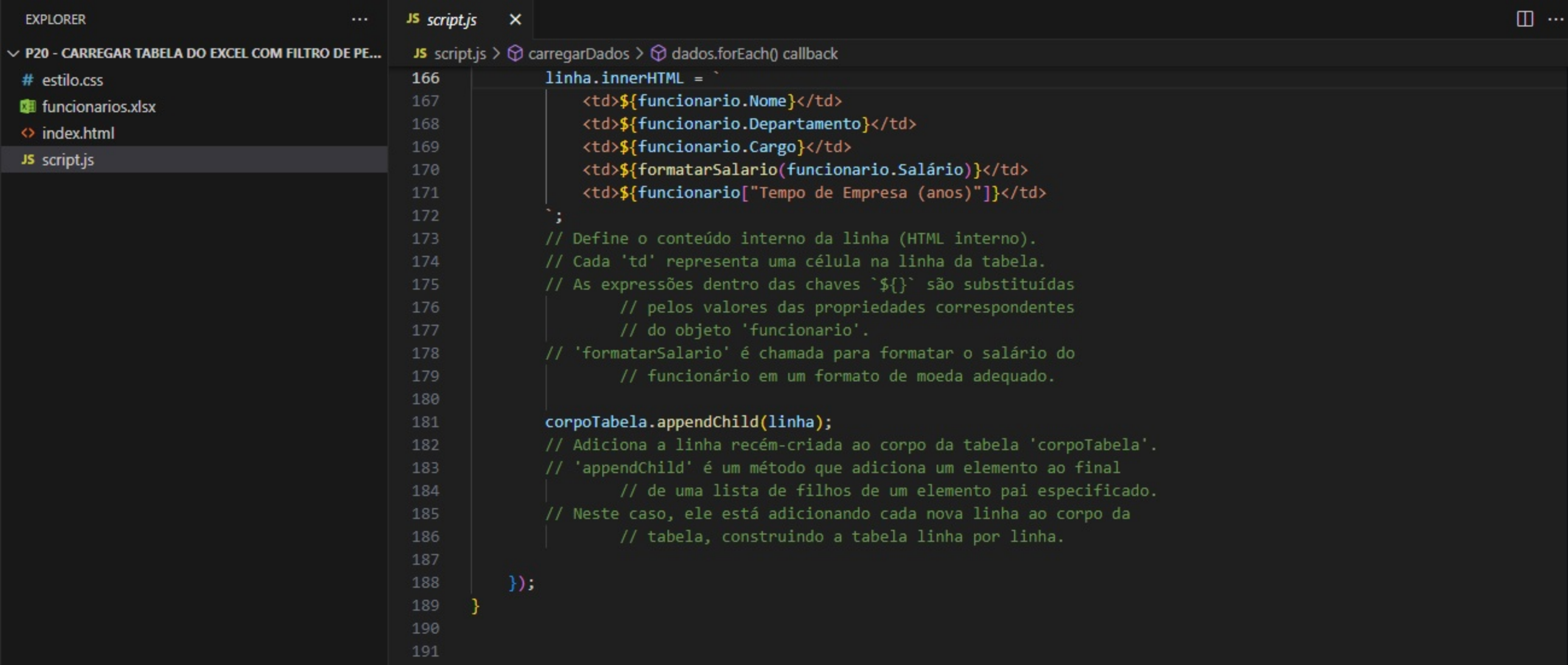

EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
script.js
script.js > filtrarTabela
var filtroTempo = document.getElementById("filtroTempo").value.toLowerCase();
// Similar ao anterior, mas para o campo de entrada do tempo
// de empresa. Permite filtrar as linhas da tabela pelo
// tempo de empresa em anos.
var linhas = document.getElementById("corpoTabela").rows;
// Acessa o elemento tbody da tabela pelo seu ID 'corpoTabela' e
// obtém todas as suas linhas (`tr`) como uma coleção HTML.
// Este objeto será usado para iterar sobre as linhas e aplicar
// os filtros de pesquisa.
// Este loop 'for' itera sobre cada linha na tabela de
// funcionários para aplicar os filtros.
for (var i = 0; i < linhas.length; i++) {
// Acessa o texto dentro da primeira célula (Nome) da linha
// atual, converte para minúsculas para padronizar a comparação.
var nome = linhas[i].cells[0].textContent.toLowerCase();
// Acessa o texto dentro da segunda célula (Departamento) da
// linha atual e o converte para minúsculas.
var departamento = linhas[i].cells[1].textContent.toLowerCase();
// Acessa o texto dentro da terceira célula (Cargo) da
// linha atual e o converte para minúsculas.
var cargo = linhas[i].cells[2].textContent.toLowerCase();

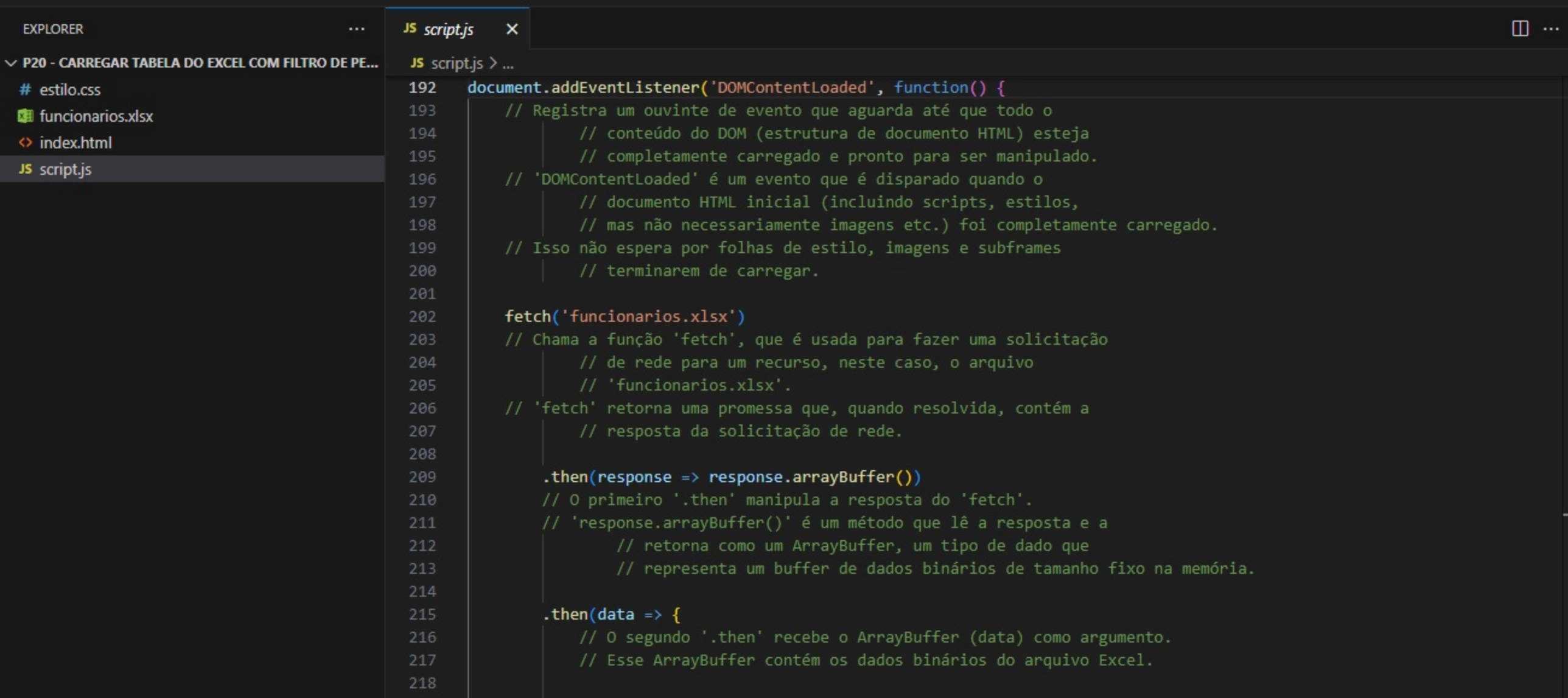

EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
script.js
script.js > filtrarTabela
// Acessa o texto dentro da quarta célula (Salário) da
// linha atual e o converte para minúsculas.
// Aqui, note que se os salários tiverem formatos especiais,
// como moeda, essa simples conversão para minúsculas
// pode não ser suficiente para comparações precisas.
var salario = linhas[i].cells[3].textContent.toLowerCase();
// Acessa o texto dentro da quinta célula (Tempo de Empresa)
// da linha atual e o converte para minúsculas.
var tempo = linhas[i].cells[4].textContent.toLowerCase();
// Define a visibilidade da linha atual com base nos
// critérios de filtro.
// A propriedade 'style.display' controla se uma
// linha é visível ou não.
linhas[i].style.display =
// Verifica se o nome na célula contém o texto
// filtrado ou se o campo de filtro está vazio.
(nome.includes(filtroNome) || filtroNome === "") &&
// Verifica se o departamento na célula contém o texto
// filtrado ou se o campo de filtro está vazio.
(departamento.includes(filtroDepartamento) || filtroDepartamento === "") &&
// Verifica se o cargo na célula contém o texto filtrado
// ou se o campo de filtro está vazio.
(cargo.includes(filtroCargo) || filtroCargo === "") &&

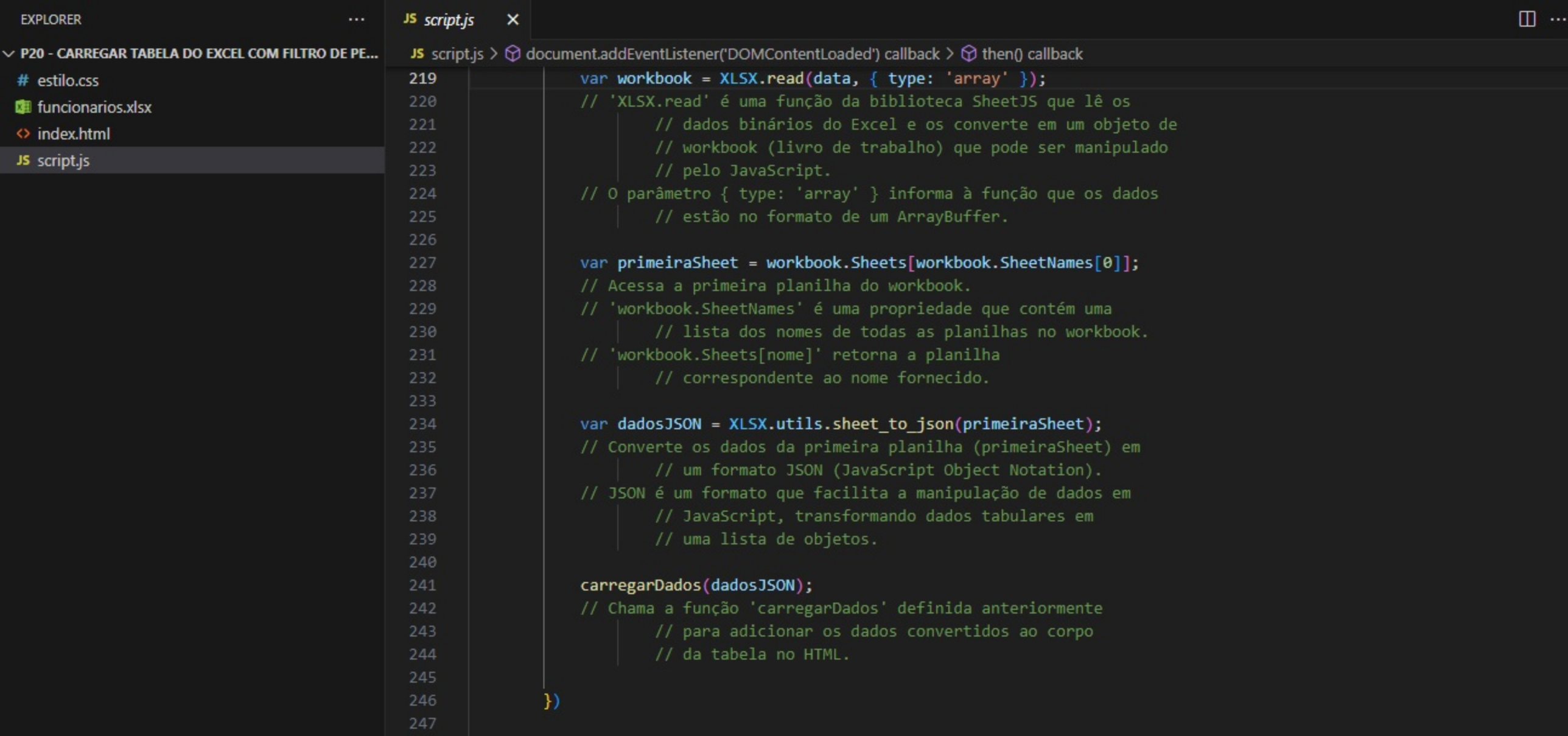

EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
script.js
script.js > filtrarTabela
// Verifica se o salário na célula contém o texto
// filtrado ou se o campo de filtro está vazio.
(salario.includes(filtroSalario) || filtroSalario === "") &&
// Verifica se o tempo de empresa na célula contém o
// texto filtrado ou se o campo de filtro está vazio.
(tempo.includes(filtroTempo) || filtroTempo === "")
? "" : "none"; // Se todas as condições forem verdadeiras,
// mostra a linha (`""` significa visível). Se alguma
// condição não for verdadeira, esconde a linha (`"none"`).
}
}
function formatarSalario(salario) {
// Definição da função 'formatarSalario'. Esta função recebe um
// parâmetro chamado 'salario', que se espera ser um valor
// numérico ou uma string que possa ser convertida em um número.
return parseFloat(salario).toLocaleString('pt-BR', { style: 'currency', currency: 'BRL' });
// O corpo da função realiza duas operações principais
// sobre o 'salario' recebido:

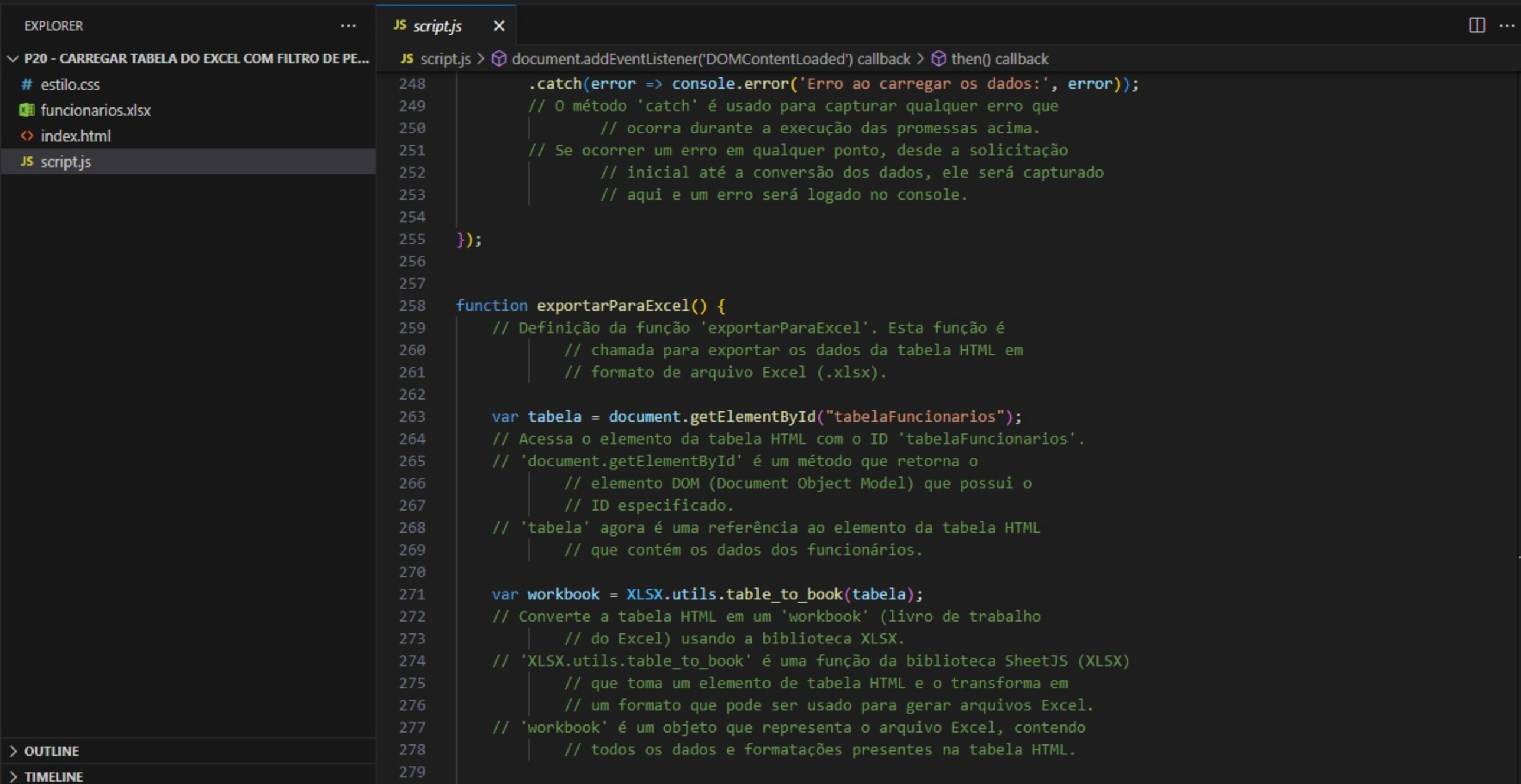

EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
script.js
script.js > formatarSalario
112 // 1. parseInt(salario):
113 // - 'parseInt' é uma função que tenta converter seu
114 // argumento para um número inteiro.
115 // - Por exemplo, se 'salario' é a string "2000.99" ou "2000",
116 // 'parseInt' irá converter isso para o número 2000.
117 // - Isso é útil para garantir que o valor a ser formatado
118 // esteja no tipo de dado correto (número).
119
120 // 2. toLocaleString('pt-BR', { style: 'currency', currency: 'BRL' }):
121 // - 'toLocaleString' é um método que formata um número para
122 // uma string de acordo com as convenções de uma
123 // localidade específica.
124 // - 'pt-BR' é o código de localidade para o Português do Brasil.
125 // - O objeto { style: 'currency', currency: 'BRL' } especifica
126 // que o número deve ser formatado como uma moeda,
127 // usando o Real Brasileiro ('BRL').
128 // - Isso transforma, por exemplo, o número 2000 em "R$ 2.000,00",
129 // que é a maneira como os valores monetários são
130 // tipicamente representados no Brasil.
131
132 // A função então retorna este valor formatado, que pode ser
133 // usado em qualquer lugar da aplicação que necessite
134 // mostrar um valor salarial de forma legível e
135 // localmente adequada.
136
137 }
138

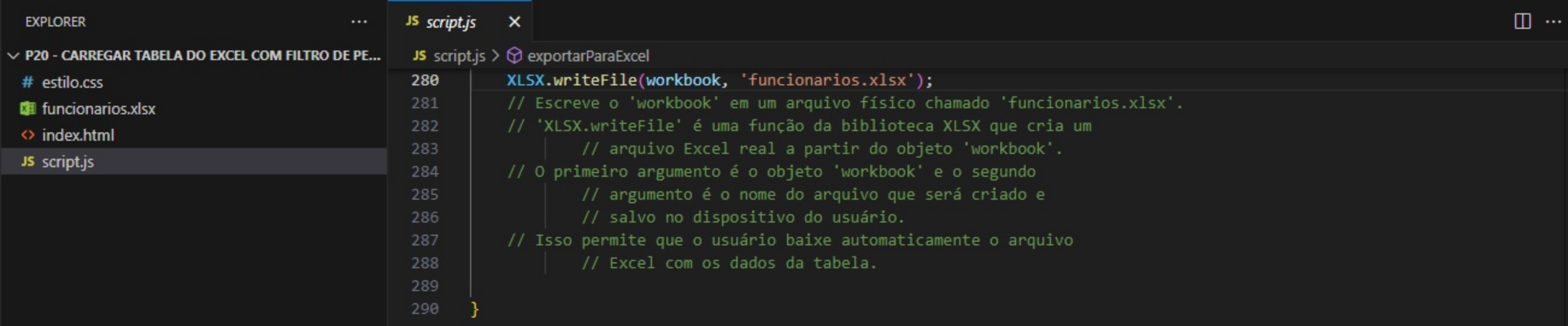

EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
script.js
script.js > carregarDados
function carregarDados(dados) {
// Define a função 'carregarDados', que é responsável por
// popular uma tabela HTML com dados fornecidos.
// O parâmetro 'dados' é esperado para ser uma lista de
// objetos, onde cada objeto representa um funcionário
// com suas informações.
var corpoTabela = document.getElementById("corpoTabela");
// Acessa o elemento tbody da tabela no HTML pelo seu ID 'corpoTabela'.
// 'document.getElementById' é uma função que retorna o
// elemento do DOM (Modelo de Objeto do Documento)
// com o ID especificado.
// 'corpoTabela' agora é uma referência ao corpo da tabela
// onde as linhas de dados serão inseridas.
dados.forEach(funcionario => {
// Inicia um loop para iterar sobre cada objeto no array 'dados'.
// 'forEach' é um método de array que executa uma função
// para cada item no array. Aqui, 'funcionario'
// representa um item individual do array 'dados'.
var linha = document.createElement("tr");
// Cria um novo elemento de linha da tabela (tr) e armazena-o
// na variável 'linha'.
// 'document.createElement' é uma função que cria um novo
// elemento HTML especificado pelo nome da tag, neste caso, "tr".


EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
script.js
script.js > carregarDados > dados.forEach() callback
linha.innerHTML = `
<td>${funcionario.Nome}</td>
<td>${funcionario.Departamento}</td>
<td>${funcionario.Cargo}</td>
<td>${formatarSalario(funcionario.Salário)}</td>
<td>${funcionario["Tempo de Empresa (anos)"]}</td>
`;
// Define o conteúdo interno da linha (HTML interno).
// Cada 'td' representa uma célula na linha da tabela.
// As expressões dentro das chaves `${}` são substituídas
// pelos valores das propriedades correspondentes
// do objeto 'funcionario'.
// 'formatarSalario' é chamada para formatar o salário do
// funcionário em um formato de moeda adequado.
corpoTabela.appendChild(linha);
// Adiciona a linha recém-criada ao corpo da tabela 'corpoTabela'.
// 'appendChild' é um método que adiciona um elemento ao final
// de uma lista de filhos de um elemento pai especificado.
// Neste caso, ele está adicionando cada nova linha ao corpo da
// tabela, construindo a tabela linha por linha.
});
}


EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
script.js
script.js > ...
192 document.addEventListener('DOMContentLoaded', function() {
193 // Registra um ouvinte de evento que aguarda até que todo o
194 // conteúdo do DOM (estrutura de documento HTML) esteja
195 // completamente carregado e pronto para ser manipulado.
196 // 'DOMContentLoaded' é um evento que é disparado quando o
197 // documento HTML inicial (incluindo scripts, estilos,
198 // mas não necessariamente imagens etc.) foi completamente carregado.
199 // Isso não espera por folhas de estilo, imagens e subframes
200 // terminarem de carregar.
201
202 fetch('funcionarios.xlsx')
203 // Chama a função 'fetch', que é usada para fazer uma solicitação
204 // de rede para um recurso, neste caso, o arquivo
205 // 'funcionarios.xlsx'.
206 // 'fetch' retorna uma promessa que, quando resolvida, contém a
207 // resposta da solicitação de rede.
208
209 .then(response => response.arrayBuffer())
210 // O primeiro '.then' manipula a resposta do 'fetch'.
211 // 'response.arrayBuffer()' é um método que lê a resposta e a
212 // retorna como um ArrayBuffer, um tipo de dado que
213 // representa um buffer de dados binários de tamanho fixo na memória.
214
215 .then(data => {
216 // O segundo '.then' recebe o ArrayBuffer (data) como argumento.
217 // Esse ArrayBuffer contém os dados binários do arquivo Excel.
218


EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
script.js
script.js > document.addEventListener('DOMContentLoaded') callback > then() callback
var workbook = XLSX.read(data, { type: 'array' });
// 'XLSX.read' é uma função da biblioteca SheetJS que lê os
// dados binários do Excel e os converte em um objeto de
// workbook (livro de trabalho) que pode ser manipulado
// pelo JavaScript.
// O parâmetro { type: 'array' } informa à função que os dados
// estão no formato de um ArrayBuffer.
var primeiraSheet = workbook.Sheets[workbook.SheetNames[0]];
// Acessa a primeira planilha do workbook.
// 'workbook.SheetNames' é uma propriedade que contém uma
// lista dos nomes de todas as planilhas no workbook.
// 'workbook.Sheets[nome]' retorna a planilha
// correspondente ao nome fornecido.
var dadosJSON = XLSX.utils.sheet_to_json(primeiraSheet);
// Converte os dados da primeira planilha (primeiraSheet) em
// um formato JSON (JavaScript Object Notation).
// JSON é um formato que facilita a manipulação de dados em
// JavaScript, transformando dados tabulares em
// uma lista de objetos.
carregarDados(dadosJSON);
// Chama a função 'carregarDados' definida anteriormente
// para adicionar os dados convertidos ao corpo
// da tabela no HTML.
})


EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
OUTLINE
TIMELINE
script.js
script.js > document.addEventListener('DOMContentLoaded') callback > then() callback
248 .catch(error => console.error('Erro ao carregar os dados:', error));
249 // O método 'catch' é usado para capturar qualquer erro que
250 // ocorra durante a execução das promessas acima.
251 // Se ocorrer um erro em qualquer ponto, desde a solicitação
252 // inicial até a conversão dos dados, ele será capturado
253 // aqui e um erro será logado no console.
254
255 });
256
257
258 function exportarParaExcel() {
259 // Definição da função 'exportarParaExcel'. Esta função é
260 // chamada para exportar os dados da tabela HTML em
261 // formato de arquivo Excel (.xlsx).
262
263 var tabela = document.getElementById("tabelaFuncionarios");
264 // Acessa o elemento da tabela HTML com o ID 'tabelaFuncionarios'.
265 // 'document.getElementById' é um método que retorna o
266 // elemento DOM (Document Object Model) que possui o
267 // ID especificado.
268 // 'tabela' agora é uma referência ao elemento da tabela HTML
269 // que contém os dados dos funcionários.
270
271 var workbook = XLSX.utils.table_to_book(tabela);
272 // Converte a tabela HTML em um 'workbook' (livro de trabalho
273 // do Excel) usando a biblioteca XLSX.
274 // 'XLSX.utils.table_to_book' é uma função da biblioteca SheetJS (XLSX)
275 // que toma um elemento de tabela HTML e o transforma em
276 // um formato que pode ser usado para gerar arquivos Excel.
277 // 'workbook' é um objeto que representa o arquivo Excel, contendo
278 // todos os dados e formatações presentes na tabela HTML.
279


EXPLORER
P20 - CARREGAR TABELA DO EXCEL COM FILTRO DE PE...
estilo.css
funcionarios.xlsx
index.html
script.js
script.js
script.js > exportarParaExcel
XLSX.writeFile(workbook, 'funcionarios.xlsx');
// Escreve o 'workbook' em um arquivo físico chamado 'funcionarios.xlsx'.
// 'XLSX.writeFile' é uma função da biblioteca XLSX que cria um
// arquivo Excel real a partir do objeto 'workbook'.
// O primeiro argumento é o objeto 'workbook' e o segundo
// argumento é o nome do arquivo que será criado e
// salvo no dispositivo do usuário.
// Isso permite que o usuário baixe automaticamente o arquivo
// Excel com os dados da tabela.
}