

UC: Projetos

DIAS: 25/10/2024

Assunto:

Small Projects – 28 a 32

UC: Projetos

DIAS: 25/10/2024

Assunto:

Small Projects

P28 - Supermercado



EXPLORER

P28 - PÁGINA SUPERMERCADO

> imagens

estilos.css

index.html

p28_supermercado.png

JS script.js



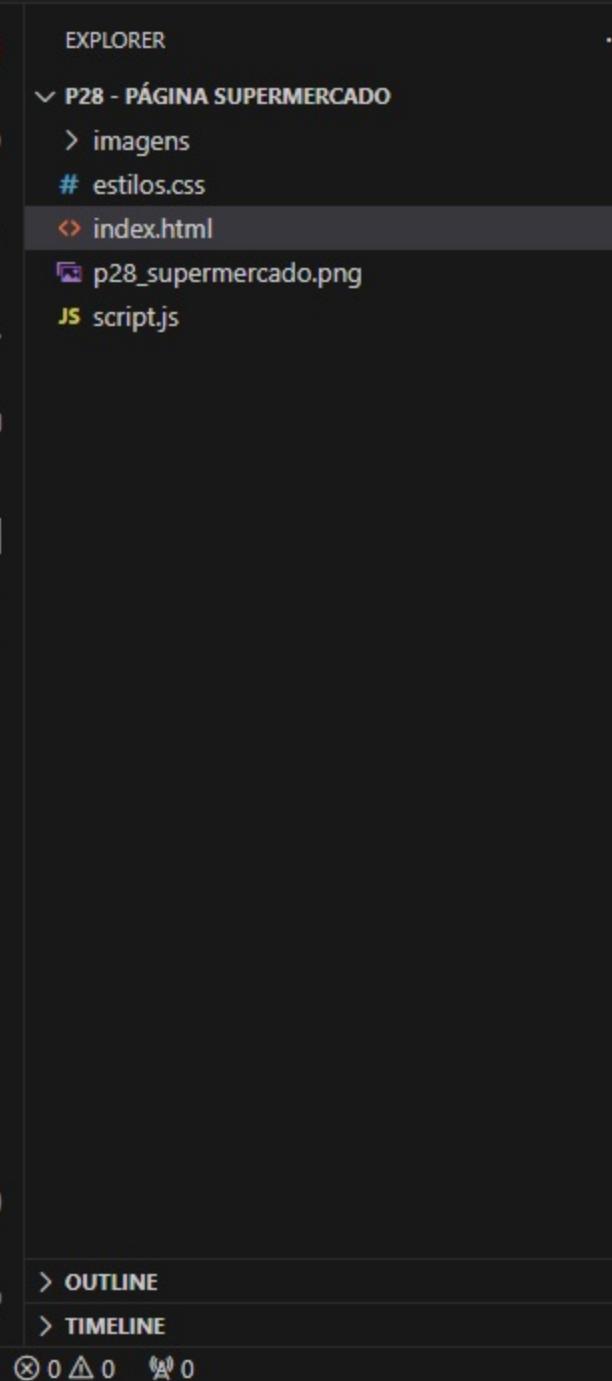
> OUTLINE

> TIMELINE

index.html X

index.html > ...

```
1  <!DOCTYPE html>
2  <!-- Declaração do tipo de documento. Aqui
3      indica que este é um documento HTML5. -->
4
5  <html lang="pt">
6  <!-- A tag <html> inicia o documento HTML. O
7      atributo 'lang="pt"' especifica que o
8      idioma principal do documento é o português. -->
9
10 <head>
11 <!-- A tag <head> contém metadados (dados sobre os dados) e
12     links para scripts e folhas de estilo. Não é
13     visível diretamente na página web. -->
14
15 <meta charset="UTF-8">
16 <!-- A tag <meta> define metadados. Aqui, 'charset="UTF-8"'
17     especifica a codificação de caracteres usada no
18     documento, que é UTF-8, capaz de representar
19     qualquer caractere do mundo. -->
20
21 <meta name="viewport" content="width=device-width, initial-scale=1.0">
22 <!-- Outra tag <meta> para controlar a visualização da
23     página em dispositivos móveis. 'width=device-width'
24     faz com que a largura da página corresponda à largura
25     da tela do dispositivo. 'initial-scale=1.0' define o
26     nível inicial de zoom quando a página é carregada
27     pela primeira vez. -->
28
29 <title>Supermercado Online</title>
30 <!-- A tag <title> define o título da página, que
31     aparece na aba do navegador. Aqui, ele é
32     definido como "Supermercado Online". -->
```



index.html

index.html > html > head

```
33 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
34     <!-- A tag &lt;link&gt; é usada para vincular a documentos
35         externos. Neste caso, está vinculando uma versão
36         minificada do Bootstrap, uma biblioteca de CSS, para
37         ajudar no design e layout da página. 'rel="stylesheet"'
38         indica que o arquivo vinculado é uma folha de estilo CSS. --&gt;
39
40     &lt;link rel="stylesheet" href="estilos.css"&gt;
41     <!-- Esta tag &lt;link&gt; vincula uma folha de estilo CSS local
42         chamada 'estilos.css'. Este arquivo conterá estilos
43         específicos para esta página. --&gt;
44
45 &lt;/head&gt;
46     <!-- Fim da seção &lt;head&gt; do documento. --&gt;
47
48 &lt;body&gt;
49     <!-- A tag &lt;body&gt; contém o conteúdo visível da página web.
50         Todo o que é exibido na janela do navegador
51         está dentro desta tag. --&gt;
52
53 &lt;header class="bg-success text-white text-center p-3 mb-3"&gt;
54     <!-- A tag &lt;header&gt; é usada para definir o cabeçalho da
55         página. Aqui, ele é estilizado com várias classes do Bootstrap:
56             - 'bg-success' aplica uma cor de fundo verde (indicativo de sucesso).
57             - 'text-white' faz com que o texto dentro do cabeçalho seja branco.
58             - 'text-center' centraliza o texto dentro do cabeçalho.
59             - 'p-3' adiciona um padding (espacamento interno) de
56                 tamanho 3 ao redor do conteúdo.
60             - 'mb-3' adiciona uma margem inferior de tamanho 3. --&gt;
61
62 &lt;h1&gt;Supermercado Online&lt;/h1&gt;
63
64</pre>
```

File Edit Selection View Go Run ... ← → ⚡ P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

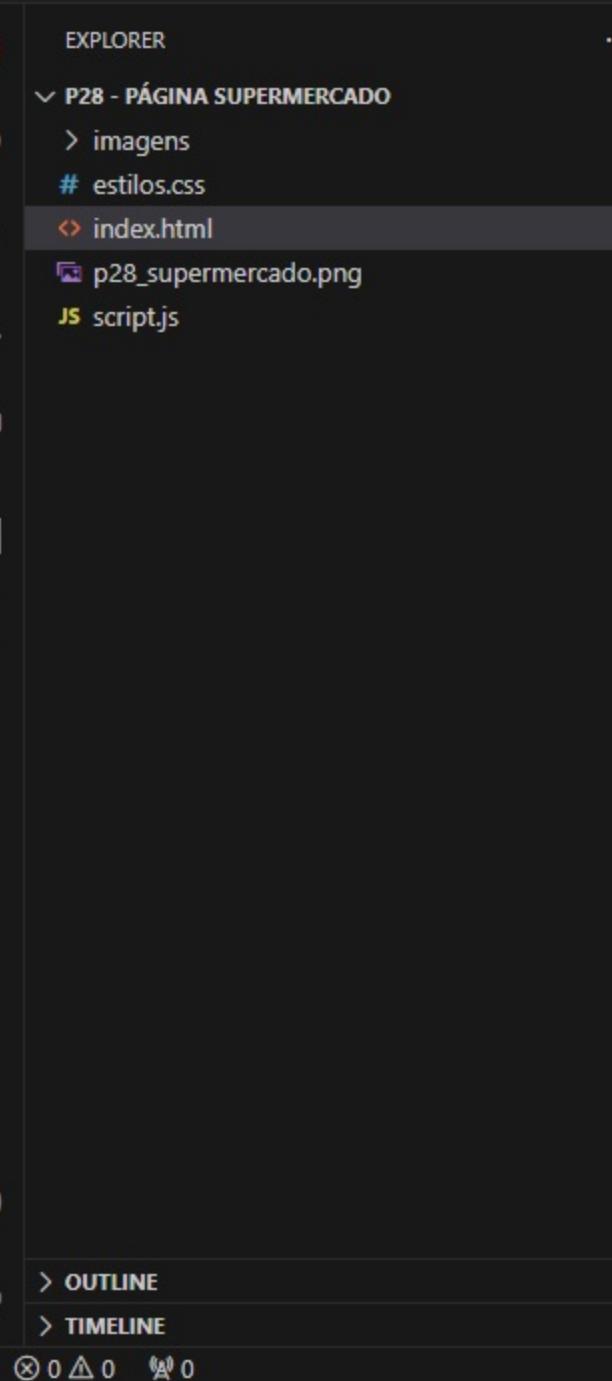
- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- script.js

index.html

```
index.html > html > body > header.bg-success.text-white.text-center.p-3.mb-3
65      <!-- A tag &lt;h1&gt; define o título principal da página. Este é o
66          título mais importante e é bom para
67          SEO (Search Engine Optimization). Aqui, ele
68          diz "Supermercado Online". --&gt;
69
70      &lt;/header&gt;
71      <!-- Fim do cabeçalho da página --&gt;
72
73      &lt;div class="container-fluid"&gt;
74          <!-- A classe 'container-fluid' é uma classe do Bootstrap que
75              cria um contêiner que abrange 100% da largura da
76              janela em todos os tamanhos de tela. --&gt;
77
78          &lt;div class="row"&gt;
79              <!-- A classe 'row' é usada para criar uma linha horizontal
80                  que pode conter colunas. É uma parte fundamental do
81                  sistema de grid do Bootstrap para layouts responsivos. --&gt;
82
83          &lt;nav class="col-md-3 col-lg-2 d-md-block sidebar bg-azul-escuro"&gt;
84              <!-- A tag &lt;nav&gt; define uma seção de navegação. Está
85                  sendo usada aqui para a barra lateral (sidebar).
86                  - 'col-md-3' define que a barra lateral ocupa 3 colunas do
87                      grid em telas médias (desktops).
88                  - 'col-lg-2' define que a barra lateral ocupa 2
89                      colunas em telas grandes.
90                  - 'd-md-block' faz com que a barra lateral seja
91                      tratada como um bloco em telas médias e maiores.
92                  - 'sidebar' é uma classe customizada
93                      para estilos específicos.
94                  - 'bg-azul-escuro' é uma classe customizada para aplicar
95                      uma cor de fundo azul escuro. --&gt;
96</pre>

Ln 65, Col 9 Spaces: 8 UTF-8 CRLF HTML ⚡ Go Live


```



index.html

```
<div class="position-sticky">
    <!-- 'position-sticky' é uma classe do CSS que faz com que
        um elemento se torne "fixo" na tela à medida que
        você rola, mas apenas dentro de seu contêiner pai. -->

    <ul class="nav flex-column">
        <!-- 'nav' e 'flex-column' são classes do Bootstrap
            usadas para definir uma lista de navegação que se
            organiza em uma coluna vertical. -->

        <li class="nav-item">
            <!-- 'nav-item' é uma classe do Bootstrap para
                itens em uma lista de navegação. -->

            <button class="nav-link active btn btn-link text-branco" id="limpar-filtro"
                    style="font-size: 26px;">
                <!-- Um botão dentro de um item da lista de navegação.
                    - 'nav-link' indica que o botão serve como um
                        link de navegação.
                    - 'active' indica que o botão está ativo.
                    - 'btn' e 'btn-link' aplicam estilos de botão
                        com aparência de link.
                    - 'text-branco' é uma classe customizada para
                        fazer o texto do botão ser branco.
                    - 'id="limpar-filtro"' é um identificador único
                        para o botão, usado por scripts.
                    - style="font-size: 26px; Aumenta o tamanho da fonte-->
                Mostrar Todos
            </button>
            <!-- Texto dentro do botão, oferecendo uma
                ação para mostrar todos os itens. -->
        </li>
    </ul>
</div>
```



EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js

index.html X

```
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
```

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

```
</li>
<!-- Fim do primeiro item da lista de navegação --&gt;

&lt;li class="nav-item"&gt;
<!-- Outro item de navegação na lista --&gt;

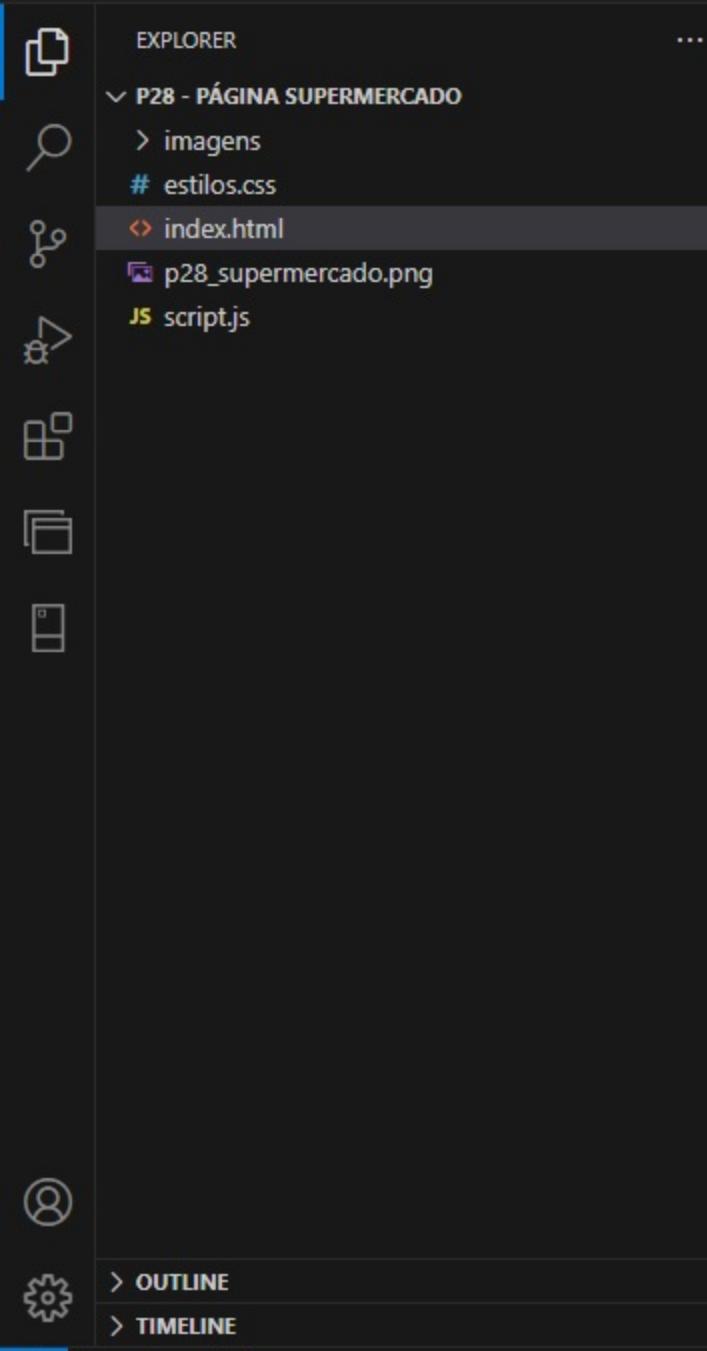
    &lt;button class="nav-link btn btn-link text-branco categoria" data-categoria="Alimentos"
    style="font-size: 24px;"&gt;
        <!-- Outro botão com estilos semelhantes ao anterior, mas
            com um atributo de dados adicionado:
            - 'data-categoria="Alimentos"' é um atributo de
            dados que pode ser usado por scripts para
            identificar que este botão está associado à
            categoria 'Alimentos'. --&gt;

        Alimentos

    &lt;/button&gt;
        <!-- Texto dentro do botão que indica a
            categoria que ele representa. --&gt;

&lt;/li&gt;
<!-- Fim do item de navegação para a categoria 'Alimentos' --&gt;

&lt;li class="nav-item"&gt;
    &lt;!-- &lt;li&gt; representa um item de lista dentro de uma
        lista não ordenada &lt;ul&gt;.
        - 'nav-item' é uma classe do Bootstrap que estiliza o
        item da lista como um componente de uma
        barra de navegação. --&gt;</pre>
```



File Edit Selection View Go Run ... ← → 🔍 P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- script.js

index.html

```
body > div.container-fluid > div.row > nav.col-md-3.col-lg-2.d-md-block.sidebar.bg-azul-escuro > div.position-sticky > ul.nav.flex-column > li.nav-item
189          Laticínios
190          |<!-- Texto do botão, indicando que ele se refere à
191          |      categoria 'Laticínios'. --&gt;
192          |      &lt;/button&gt;
193          |  &lt;/li&gt;
194          |  |<!-- Fim do item da lista de navegação para a
195          |      |      categoria 'Laticínios' --&gt;
196
197
198          |&lt;li class="nav-item"&gt;
199          |  |<!-- Mais um item de lista para uma outra categoria de produtos. --&gt;
200
201          |          &lt;button class="nav-link btn btn-link text-branco categoria" data-categoria="Produtos de
202          |              Limpeza" style="font-size: 24px;"&gt;
203          |              |<!-- Botão para a categoria 'Produtos de Limpeza',
204          |                  |      novamente com as mesmas classes e um
205          |                  |      atributo de dados específico. --&gt;
206
207          |          Produtos de Limpeza
208          |          |<!-- Texto no botão que mostra qual categoria
209          |              |      ele representa. --&gt;
210          |          &lt;/button&gt;
211          |      &lt;/li&gt;
212          |      |<!-- Fim do item da lista de navegação para a
213          |          |      categoria 'Produtos de Limpeza' --&gt;
214
215          |&lt;li class="nav-item"&gt;
216          |  |<!-- Item de lista para a última categoria mencionada. --&gt;
217
218          |          &lt;button class="nav-link btn btn-link text-branco categoria" data-categoria="Snacks"
219          |              style="font-size: 24px;"&gt;</pre>

Ln 189, Col 1 Spaces: 8 UTF-8 CRLF HTML Go Live


```

File Edit Selection View Go Run ... ← → P28 - Página Supermercado

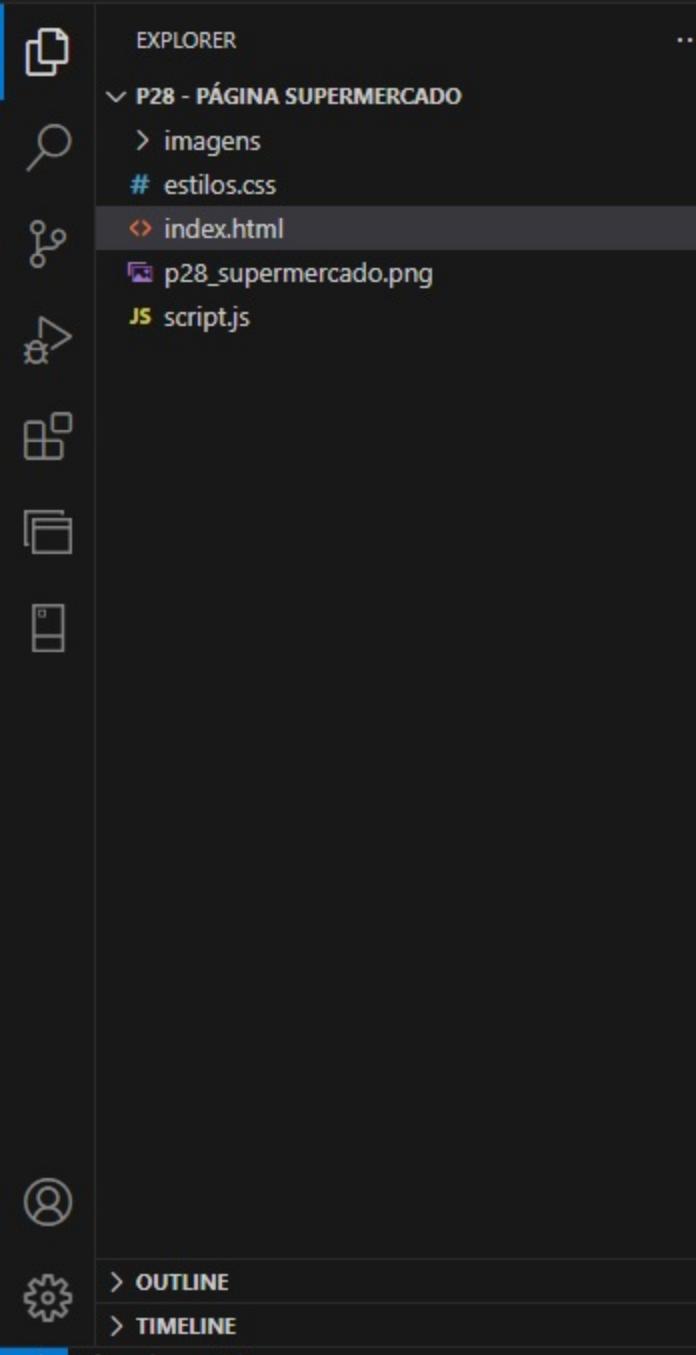
EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- script.js

index.html

```
ol-md-3.col-lg-2.d-md-block.sidebar.bg-azul-escuro > div.position-sticky > ul.nav.flex-column > li.nav-item > button.nav-link.btn.btn-link.text-branco.categoria
219                                         |<!-- Botão para a categoria 'Snacks', estilizado da mesma
220                                         |                                           maneira que os outros botões. --&gt;
221                                         |Snacks
222                                         |<!-- Texto no botão que identifica a categoria de
223                                         |                                           produtos que ele representa. --&gt;
224                                         |                                         </button>
225                                         |                                         
226                                         |                                         
227                                         |                                         
228                                         |                                         
229                                         |                                         
230                                         
231                                         
232                                         
233                                         
234                                         
```



EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- script.js

OUTLINE

TIMELINE

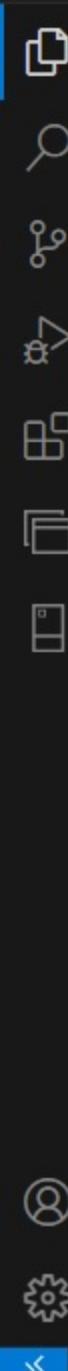
index.html X

ex.html > html > body > div.container-fluid > div.row > main.col-md-9.ms-sm-auto.col-lg-10.px-md-4 > div.row.mb-3 > div.col > div.input-group

```
282 <button class="btn btn-outline-secondary" type="button" id="botao-pesquisar" style="font-size: 24px;">Pesquisar</button>
283     <!-- &lt;button&gt; cria um botão clicável.
284         - 'btn' e 'btn-outline-secondary' são classes do Bootstrap que
285             estilizam o botão com um contorno discreto.
286         - 'type="button"' especifica que o botão não tem um
287             comportamento padrão de submissão de formulário.
288         - 'id="botao-pesquisar"' fornece um identificador único para o
289             botão, útil para interações de JavaScript. --&gt;
290
291     &lt;/div&gt;
292     <!-- Fim do grupo de entrada --&gt;
293
294 &lt;/div&gt;
295     <!-- Fim da primeira coluna --&gt;
296
297
298 &lt;div class="col text-end"&gt;
299     <!-- Outra 'col' que ocupa o espaço restante na 'row'. 'text-end'
300         alinha o texto ou conteúdo ao final (direita) da coluna. --&gt;
301
302     &lt;button class="btn btn-info" id="botao-ver-carrinho" data-bs-toggle="modal" data-bs-target="#modalCarrinho" style="font-size: 24px;"&gt;
303         <!-- Cria outro botão.
304             - 'btn btn-info' aplica o estilo do Bootstrap com uma cor
305                 informativa (geralmente azul).
306             - 'id="botao-ver-carrinho"' é um identificador único.
307             - 'data-bs-toggle="modal"' e 'data-bs-target="#modalCarrinho"'
308                 são atributos de dados do Bootstrap para controlar a
309                 exibição de um modal, especificamente o modal
310                 com id 'modalCarrinho'.
311             - style="font-size: 24px; aumenta o tamanho da fonte"--&gt;</pre>

Ln 282, Col 1 Spaces: 8 UTF-8 CRLF HTML Go Live


```



EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js

index.html X

```
ex.html > html > body > div.container-fluid > div.row > main.col-md-9.ms-sm-auto.col-lg-10.px-md-4 > div.row.mb-3 > div.col.text-end > button#ver_carrinho

312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343

    Ver Carrinho (<span id="contador-carrinho">0</span> itens)
    <!-- Texto dentro do botão com um &lt;span&gt; embutido que mostra o
        número de itens no carrinho. 'id="contador-carrinho"'
        permite que esse número seja atualizado
        dinamicamente com JavaScript. --&gt;
    &lt;/button&gt;
&lt;/div&gt;
<!-- Fim da segunda coluna --&gt;

&lt;/div&gt;
<!-- Fim da 'row' --&gt;

&lt;div id="secoes-produtos"&gt;
    <!-- Um &lt;div&gt; para conter as seções de produtos. 'id="secoes-produtos"'
        permite que este &lt;div&gt; seja referenciado por JavaScript
        para adicionar dinamicamente conteúdo ao documento. --&gt;
    <!-- As seções de produtos serão adicionadas aqui
        pelo JavaScript --&gt;
&lt;/div&gt;
<!-- Fim do contêiner de seções de produtos --&gt;

&lt;/main&gt;
<!-- Fim da tag &lt;main&gt;, que contém o conteúdo
    principal da página. --&gt;

&lt;/div&gt;
<!-- Fim da 'div' que continha a linha principal --&gt;</pre>
```

File Edit Selection View Go Run ... ← → ⌂ P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- script.js

index.html

```
345     <!-- Fim da 'div' que continha o contêiner fluido -->
346
347
348     <!-- Modal para exibir itens do carrinho -->
349     <!-- Este comentário descreve a função do código a seguir, que é
350         criar um modal para exibir os itens do carrinho
351         de compras. -->
352
353     <div class="modal fade" id="modalCarrinho" tabindex="-1" aria-labelledby="labelModalCarrinho" aria-hidden="true">
354         <!-- <div> cria um contêiner para o modal.
355             - 'modal fade' são classes do Bootstrap. 'modal' define o
356                 elemento como um modal e 'fade' adiciona uma
357                     transição de desvanecimento ao abrir e fechar.
358             - 'id="modalCarrinho"' fornece um identificador único
359                 para o modal, permitindo referenciá-lo
360                     especificamente em scripts ou estilos.
361             - 'tabindex="-1"' permite que o modal seja focado para
362                 acessibilidade e interações do teclado, mas não é
363                     focalizável através de navegação sequencial do teclado.
364             - 'aria-labelledby="labelModalCarrinho"' associa o modal a um
365                 título para acessibilidade, indicando que o elemento
366                     que possui o ID 'labelModalCarrinho' serve como o rótulo.
367             - 'aria-hidden="true"' informa às tecnologias assistivas que o
368                 modal está inicialmente oculto. -->
369
370     <div class="modal-dialog">
371         <!-- 'modal-dialog' é uma classe do Bootstrap que encapsula o
372             diálogo/modal propriamente dito, centrando-o e aplicando
373                 estilos necessários para a caixa de diálogo. -->
374
375         <div class="modal-content">
376             <!-- 'modal-content' define o conteúdo dentro do modal,
377                 ...
378             </div>
379     </div>
380 </div>
```

Ln 345, Col 1 Spaces: 8 UTF-8 CRLF HTML Go Live

File Edit Selection View Go Run ... ← → P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- imagens
- # estilos.css
- index.html
- p28_supermercado.png
- script.js

index.html

```
<div class="modal-header">
    <!-- 'modal-header' é uma seção do modal que contém o
        cabeçalho, geralmente incluindo um título e um
        botão para fechar o modal. -->

    <h5 class="modal-title" id="labelModalCarrinho">Carrinho de Compras</h5>
    <!-- <h5> define o título do modal.
        - 'modal-title' é uma classe que estiliza o
            título dentro do cabeçalho do modal.
        - 'id="labelModalCarrinho"' é usado em conjunto
            com 'aria-labelledby' no contêiner do
            modal para acessibilidade. -->

    <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
    <!-- <button> cria um botão para fechar o modal.
        - 'btn-close' é uma classe do Bootstrap que
            aplica um ícone de fechamento.
        - 'data-bs-dismiss="modal"' é um atributo de
            dados do Bootstrap que aciona o fechamento do modal.
        - 'aria-label="Close"' fornece uma descrição
            acessível para o botão de fechar, importante
            para usuários de leitores de tela. -->

</div>
<!-- Fim do cabeçalho do modal -->

<div class="modal-body">
    <!-- 'modal-body' contém o corpo principal do modal,
        onde o conteúdo dinâmico é geralmente inserido. -->
```

Ln 377, Col 8 Spaces: 8 UTF-8 CRLF HTML Go Live

File Edit Selection View Go Run ... ← → P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- imagenes
- # estilos.css
- index.html
- p28_supermercado.png
- script.js

index.html

```
<ul id="lista-itens-carrinho" class="list-group">
    <!-- <ul> define uma lista não ordenada.
        - 'id="lista-itens-carrinho"' fornece um identificador
            único para a lista, usado em scripts para
            adicionar itens ao carrinho.
        - 'list-group' é uma classe do Bootstrap que
            estiliza a lista para melhor visualização e
            organização. -->
    <!-- Itens do carrinho serão adicionados aqui
        pelo JavaScript -->
    <!-- Este comentário indica que o conteúdo da
        lista será gerenciado e inserido
        dinamicamente por JavaScript. -->

</ul>
</div>
<!-- Fim do corpo do modal -->

<div class="modal-footer">
    <!-- 'modal-footer' é uma seção do modal que contém o
        rodapé, tipicamente usado para botões de
        ação como 'Fechar' ou 'Salvar'. -->

    <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Fechar</button>
    <!-- Outro <button> para fechar o modal.
        - 'btn btn-secondary' são classes do Bootstrap que
            estilizam o botão com um esquema de
            cores secundário.
        - 'data-bs-dismiss="modal"' permite que o botão
            feche o modal quando clicado. -->

```

Ln 409, Col 4 Spaces: 8 UTF-8 CRLF HTML Go Live



EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js



index.html X

```
↳ index.html > html > body > div#modalCarrinho.modal.fade > div.modal-dialog > div.modal-content > div.modal-footer
441           
```

442 </div>
443 <!-- Fim do rodapé do modal -->
444

```
445           
```

446

```
447           </div>  
448           <!-- Fim do modal geral --&gt;<br/>449  
450           <!-- Modal para exibir detalhes do produto -->  
451           <!-- Este comentário descreve a função do código a seguir,  
452               que é criar um modal para exibir detalhes de  
453               um produto específico. -->  
454  
455  
456           <div class="modal fade" id="modalProduto" tabindex="-1" aria-labelledby="labelModalProduto" aria-hidden="true">  
457           <!-- <div> cria um contêiner para o modal.  
458               - 'modal fade' são classes do Bootstrap que definem o  
459                   elemento como um modal e aplicam uma transição  
460                   de desvanecimento.  
461               - 'id="modalProduto"' fornece um identificador único  
462                   para o modal, útil para referências específicas  
463                   em scripts ou CSS.  
464               - 'tabindex="-1"' permite que o modal seja focado  
465                   para interações do teclado, mas não através de  
466                   navegação sequencial do teclado.  
467               - 'aria-labelledby="labelModalProduto"' associa o  
468                   modal ao título para acessibilidade.  
469               - 'aria-hidden="true"' indica que o modal está  
470                   inicialmente oculto para tecnologias assistivas. -->  
471  
472           <div class="modal-dialog">
```



> OUTLINE

> TIMELINE



⊗ 0 ▲ 0 ⌂ 0

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js

OUTLINE

TIMELINE

index.html

```
index.html > html > body > div#modalProduto.modal.fade > div.modal-dialog
473    <!-- 'modal-dialog' é uma classe do Bootstrap que
474        encapsula o diálogo/modal propriamente dito,
475        centrando-o e aplicando estilos necessários. -->
476
477    <div class="modal-content">
478        <!-- 'modal-content' define o conteúdo interno do
479            modal, incluindo cabeçalho, corpo e rodapé
480            com um fundo e bordas. -->
481
482        <div class="modal-header">
483            <!-- 'modal-header' contém o cabeçalho do modal,
484                geralmente inclui um título e um botão
485                para fechar o modal. -->
486
487            <h5 class="modal-title" id="labelModalProduto">Detalhes do Produto</h5>
488            <!-- <h5> define o título do modal. 'modal-title'
489                estiliza o título dentro do cabeçalho do modal. -->
490
491            <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
492            <!-- <button> cria um botão para fechar o modal. 'btn-close'
493                aplica um ícone de fechamento, e 'data-bs-dismiss="modal"'
494                aciona o fechamento do modal. 'aria-label="Close"'
495                fornece uma descrição acessível para o botão. -->
496
497        </div>
498        <!-- Fim do cabeçalho do modal -->
499
500        <div class="modal-body">
501            <!-- 'modal-body' é a seção que contém o corpo principal do
502                modal, onde o conteúdo detalhado é inserido. -->
503
504            <div class="text-center">
```

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js

OUTLINE

TIMELINE

index.html

```
index.html > html > body > div#modalProduto.modal.fade > div.modal-dialog > div.modal-content > div.modal-body > div.text-center
505      <!-- 'text-center' é uma classe do Bootstrap que
506          centraliza o conteúdo dentro deste <div>. -->
507
508      <img id="imagem-produto" class="img-fluid mb-3" src="" alt=""/>
509      <!-- <img> exibe uma imagem.
510          - 'id="imagem-produto"' é um identificador
511              único para a imagem.
512          - 'img-fluid' é uma classe do Bootstrap que
513              torna a imagem responsiva.
514          - 'mb-3' adiciona uma margem inferior.
515          - 'src=""' será o caminho para a imagem do
516              produto, definido dinamicamente.
517          - 'alt=""' deve conter um texto alternativo que
518              descreva a imagem, importante para acessibilidade. -->
519      </div>
520
521      <h5 id="nome-produto"></h5>
522      <!-- <h5> mostra o nome do produto, o conteúdo
523          será inserido dinamicamente. -->
524
525      <p id="descricao-produto"></p>
526      <!-- <p> mostra a descrição do produto, também
527          preenchida dinamicamente. -->
528
529      <p><strong>Preço: R$ <span id="preco-produto"></span></strong></p>
530      <!-- Exibe o preço do produto, com o valor sendo
531          adicionado dinamicamente no <span>. -->
532
533      <div class="input-group mb-3">
534          <!-- 'input-group' é uma classe do Bootstrap usada
535              para agrupar um <input> com um <label> ou
536              botão para formar um campo de entrada estilizado. -->
```

File Edit Selection View Go Run ... ← → P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- imagens
- # estilos.css
- index.html
- p28_supermercado.png
- script.js

index.html

```
537 <label class="input-group-text" for="quantidade-produto">Quantidade</label>
538 <!-- <label> define um rótulo para o campo de
539     entrada seguinte, 'input-group-text' estiliza o
540         rótulo como parte do grupo de entrada. -->
541
542 <input type="number" class="form-control" id="quantidade-produto" value="1" min="1">
543 <!-- <input type="number"> cria um campo de entrada para números.
544     - 'form-control' estiliza o campo.
545     - 'id="quantidade-produto"' é um identificador único.
546     - 'value="1"' define o valor inicial como 1.
547     - 'min="1"' assegura que o valor não seja menor que 1. -->
548
549 </div>
550
551 <button class="btn btn-primary" id="botao-adicionar-carrinho-modal" data-produto-id="">
552 <!-- <button> cria um botão para adicionar o
553     produto ao carrinho.
554     - 'btn btn-primary' aplica estilos do Bootstrap para
555         um botão primário (geralmente azul).
556     - 'id="botao-adicionar-carrinho-modal"' é um
557         identificador único.
558     - 'data-produto-id=""' é um atributo de dados que
559         será preenchido com o ID do produto para
560             identificação ao adicionar ao carrinho. -->
561
562     Adicionar ao Carrinho
563
564 </button>
565 </div>
566 <!-- Fim do corpo do modal -->
567
568 </div>
```

Ln 537, Col 5 Spaces: 8 UTF-8 CRLF HTML Go Live

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js

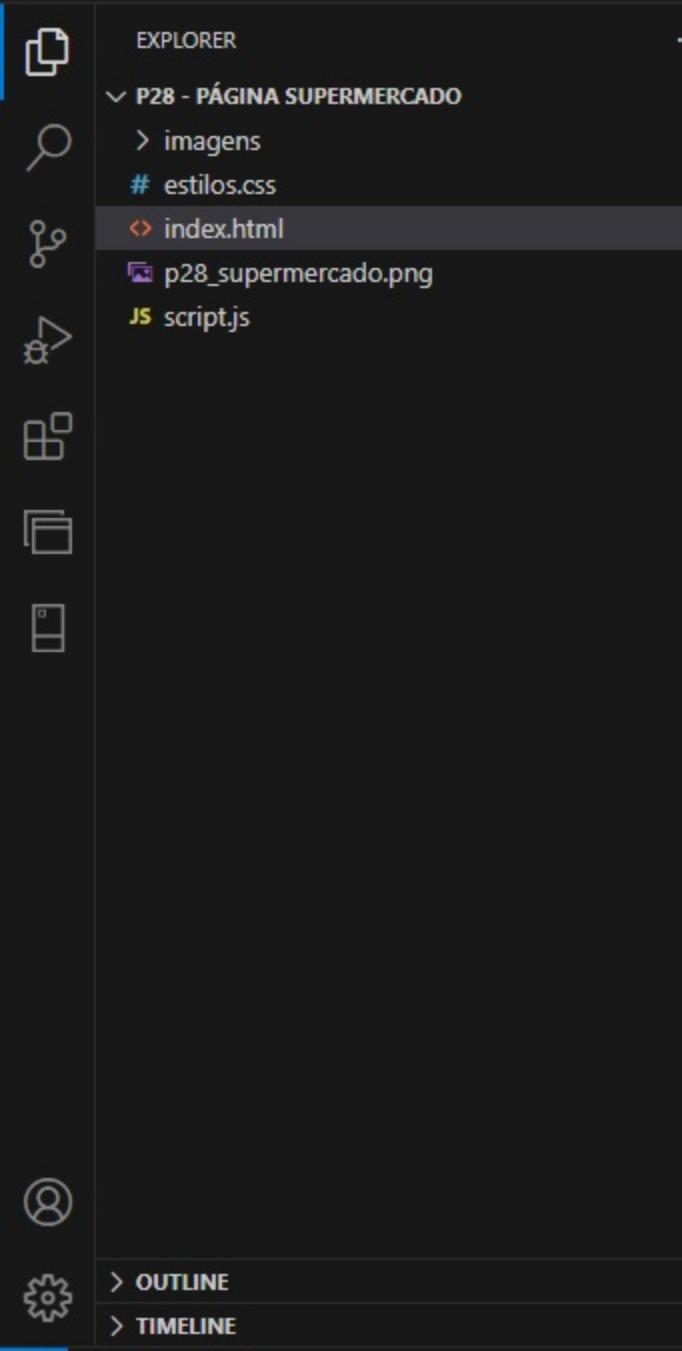
OUTLINE

TIMELINE

index.html X

index.html > html > body > div#modalProduto.modal.fade > div.modal-dialog

```
569     </div>
570     </div>
571     <!-- Fim do modal geral -->
572
573
574     <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
575     <!-- A tag <script> é usada para incorporar ou referenciar um
576         script executável, geralmente JavaScript.
577         'src="https://code.jquery.com/jquery-3.6.0.min.js"' especifica o
578         caminho para a biblioteca jQuery. jQuery é uma biblioteca
579         JavaScript rápida e concisa que simplifica a manipulação de
580         documentos HTML, o tratamento de eventos, animações e
581         interações Ajax para desenvolvimento web rápido.
582         Esta versão específica (3.6.0) é uma versão minimizada ('min.js'), o
583         que significa que o código foi compactado para reduzir o
584         tamanho do arquivo, acelerando o tempo de carregamento
585         da página. -->
586
587     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
588     <!-- Outra tag <script> para incluir o Bootstrap, uma biblioteca de
589         componentes front-end para desenvolver websites e aplicações web.
590         'src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"' aponta
591         para a versão minimizada do Bootstrap, que inclui tanto o Bootstrap
592         quanto as dependências de popper.js necessárias para tooltips e popovers.
593         'bootstrap.bundle.min.js' inclui todos os plugins JavaScript de Bootstrap
594         juntamente com a biblioteca Popper necessária para certos componentes
595         como modais, tooltips, etc. -->
596
597     <script src="script.js"></script>
598     <!-- Esta tag <script> referencia um arquivo JavaScript
599         local chamado 'script.js'.
600         'src="script.js"' especifica que o arquivo está localizado na
```



index.html X

index.html > html > body

```
601      mesma pasta que o documento HTML ou de acordo com a
602      estrutura especificada no caminho do arquivo.
603      - Este arquivo contém o código JavaScript
604      específico que interage com a página HTML, como
605      manipulação de eventos, interações dinâmicas com o
606      usuário, e outras funções específicas da lógica
607      da aplicação. -->
608
609  </body>
610  <!-- Fim da tag &lt;body&gt;, que contém todo o conteúdo
611      visível e scripts da página. --&gt;
612
613  &lt;/html&gt;
614  <!-- Fim da tag &lt;html&gt;, que encerra o documento HTML. Todos os
615      conteúdos e estruturas da página web devem estar contidos
616      entre a abertura e o fechamento desta tag. --&gt;</pre>
```



EXPLORER

P28 - PÁGINA SUPERMERCADO

> imagens

estilos.css

< index.html

p28_supermercado.png

JS script.js



> OUTLINE

> TIMELINE

estilos.css X

```
# estilos.css > #secoes-produtos .card
1  #secoes-produtos .card {
2
3      margin-bottom: 20px;
4      /* Define uma margem inferior de 20 pixels para os
5         elementos com a classe 'card' que estão dentro
6         do elemento com o ID 'secoes-produtos'.
7         Isso ajuda a criar espaço entre os cartões de
8         produtos quando empilhados verticalmente. */
9
10 }
11
12 .bg-imagem-topo {
13
14     width: 100%;
15     /* Define a largura da imagem para ocupar 100% do
16        elemento pai, garantindo que a imagem estenda-se
17        por toda a largura disponível. */
18
19     height: 200px;
20     /* Estabelece uma altura fixa de 200 pixels para a
21        imagem, o que proporciona uniformidade na
22        exibição das imagens em diferentes cartões. */
23
24     object-fit: contain;
25     /* Garante que toda a imagem seja visível dentro do
26        espaço definido sem distorção, ajustando a imagem
27        dentro das dimensões sem recortá-la. */
28
29     background: ■#fff;
30     /* Define a cor de fundo do elemento para
31        branco (#fff), útil se a imagem não cobrir todo o
32        espaço definido, evitando fundos inesperados. */
```

File Edit Selection View Go Run ... ← → ⌂ P28 - Página Supermercado

EXPLORER

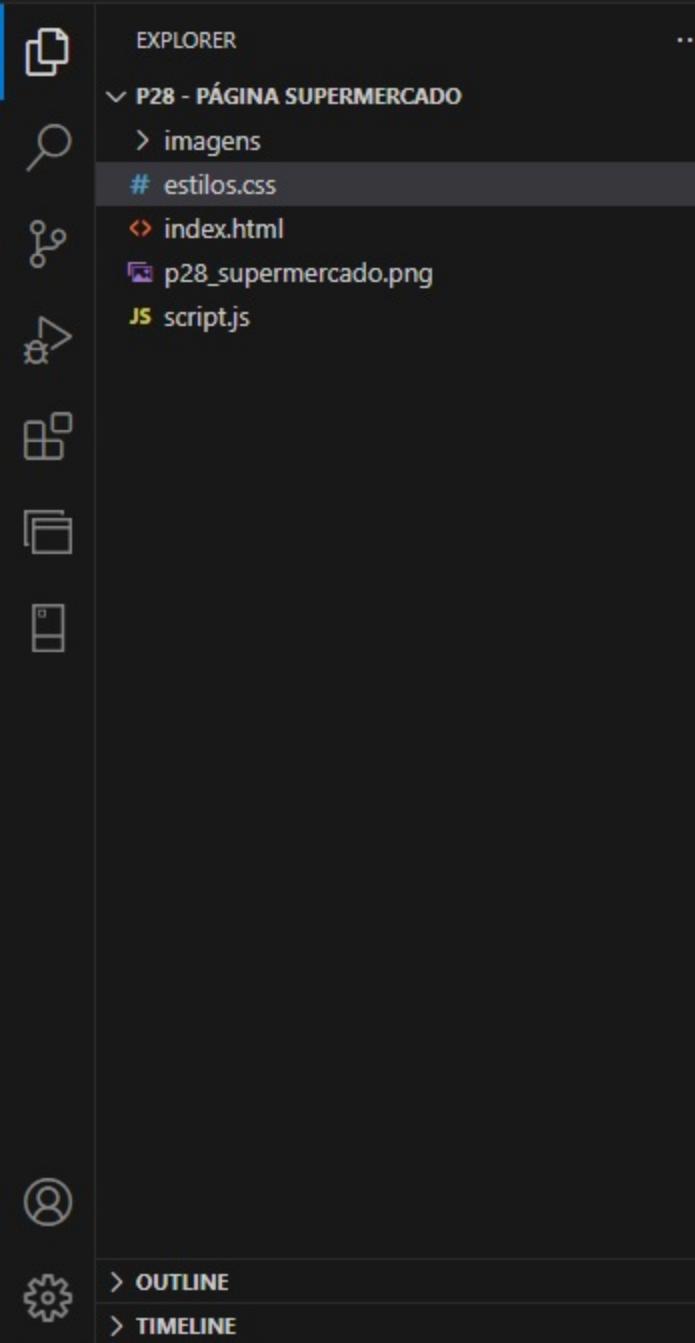
P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js

estilos.css X

```
# estilos.css > .bg-imagem-topo
33
34 }
35
36 .item-carrinho {
37
38     display: flex;
39     /* Aplica o modelo de caixa flexível, facilitando o
40      alinhamento dos itens dentro deste elemento
41      horizontalmente. */
42
43     align-items: center;
44     /* Centraliza verticalmente os itens dentro do
45      elemento, útil para alinhar itens de diferentes
46      tamanhos (como texto e imagens) na mesma linha. */
47
48 }
49
50 .imagem-item-carrinho {
51
52     width: 50px;
53     /* Define a largura da imagem do item do
54      carrinho para 50 pixels, mantendo as
55      imagens pequenas e consistentes em tamanho. */
56
57     height: 50px;
58     /* Define a altura da imagem do item do carrinho
59      para 50 pixels, assegurando que as imagens
60      sejam sempre quadradas e uniformes. */
61
62     object-fit: cover;
63     /* Garante que a imagem preencha completamente o
64      espaço definido (50x50 pixels), cortando o
```

Ln 33, Col 1 Spaces: 8 UTF-8 CRLF CSS ⌂ Go Live



```
# estilos.css X
# estilos.css > .imagem-item-carrinho
65    excesso de imagem se necessário para cobrir
66    todo o espaço. */
67
68    margin-right: 10px;
69    /* Adiciona uma margem à direita de 10 pixels,
70       separando a imagem dos textos ou outros
71       elementos ao lado dentro do carrinho. */
72 }
73
74
75
76 .detalhes-item-carrinho {
77
78     display: flex;
79     /* Configura o contêiner 'detalhes-item-carrinho'
80        para usar flexbox, que é um modelo de layout
81        projetado para estruturar elementos de forma
82        mais eficiente e com melhor controle. */
83
84     flex-direction: column;
85     /* Define a direção dos itens flex como 'column', o que
86        significa que os itens filhos (como nomes, preços, etc.)
87        serão empilhados verticalmente, um abaixo do outro. */
88
89 }
90
91 .sidebar {
92
93     position: fixed;
94     /* A propriedade 'position: fixed' é usada para fixar a
95        barra lateral na janela de visualização, fazendo
96        com que ela não se move ao rolar a página. */
```



EXPLORER

✓ P28 - PÁGINA SUPERMERCADO

```
> imagens  
# estilos.css  
<> index.html  
♫ p28_supermercado.png  
JS script.js
```

estilos.css

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Title Bar:** P28 - Página Supermercado
- Sidebar (Left):**
 - EXPLORER
 - P28 - PÁGINA SUPERMERCADO
 - > imagens
 - # estilos.css
 - index.html
 - p28_supermercado.png
 - script.js
 - OUTLINE
 - TIMELINE
- Code Editor (Main Area):**

estilos.css

```
129
130 }
131
132 .sidebar .nav-link {
133     font-weight: 500;
134     /* 'font-weight: 500' faz o texto dentro de '.nav-link' ser
135      | semi-negrito, ajudando a destacar os links na navegação. */
136
137     color: #ffffff;
138     /* 'color: #ffffff' define a cor do texto para
139      | branco, garantindo que os links sejam facilmente
140      | visíveis contra o fundo azul marinho. */
141
142     /* Aumenta o tamanho da fonte para melhor legibilidade. */
143     font-size: 18px;
144
145 }
146
147
148 .sidebar .nav-link:hover {
149
150     color: #050505;
151     /* Muda a cor do texto para quase preto quando o
152      | mouse passa sobre o link, aumentando a visibilidade. */
153
154     background-color: #rgb(182, 209, 123);
155     /* Altera a cor de fundo para um verde claro,
156      | proporcionando um feedback visual
157      | claro de interatividade. */
158
159     width: 100%;
160     /* Garante que o fundo no hover estenda-se
```
- Bottom Status Bar:** Ln 129, Col 1, Spaces: 8, UTF-8, CRLF, CSS, Go Live



EXPLORER

P28 - PÁGINA SUPERMERCADO

> imagens

estilos.css

<> index.html

p28_supermercado.png

JS script.js

JS script.js

JS script.js > ...

```
1  document.addEventListener("DOMContentLoaded", function() {  
2      // Este evento 'DOMContentLoaded' é acionado  
3      // quando todo o conteúdo HTML foi completamente carregado,  
4      // sem esperar pelo CSS, imagens ou iframes para  
5      // finalizar. É o ponto onde é seguro manipular o DOM.  
6  
7  let carrinho = JSON.parse(localStorage.getItem('carrinho')) || [];  
8  // Tenta recuperar o array 'carrinho' do  
9  // armazenamento local (localStorage).  
10 // 'JSON.parse' converte a string JSON armazenada de  
11 // volta para um objeto JavaScript.  
12 // Se 'localStorage.getItem('carrinho')' retorna  
13 // null (carrinho não existe), o operador '||' garante  
14 // que 'carrinho' será um array vazio.  
15  
16 const categorias = {  
17  
18    // Declara um objeto 'categorias' que contém arrays  
19    // de objetos para cada categoria de produtos.  
20  "Alimentos": [  
21  
22    // Array de objetos representando produtos na  
23    // categoria "Alimentos".  
24    { id: 1, nome: "Arroz", descricao: "Arroz branco longo, 5kg", imageUrl: "imagens/arroz.jpg", preco: "19,90" },  
25    // Cada objeto tem um 'id' único, 'nome', 'descricao', 'imageUrl'  
26    // para a foto do produto, e 'preco'.  
27    { id: 2, nome: "Feijão", descricao: "Feijão preto, 1kg", imageUrl: "imagens/feijao.jpg", preco: "7,50" },  
28    { id: 3, nome: "Macarrão", descricao: "Macarrão integral, 500g", imageUrl: "imagens/macarrao.jpg",  
29    preco: "4,99" }  
30  ],  
31  "Frutas": [  
32    { id: 1, nome: "Maçã", descricao: "Maçã vermelha, 1kg", imageUrl: "imagens/maca.jpg", preco: "5,00" },  
33    { id: 2, nome: "Laranja", descricao: "Laranja, 1kg", imageUrl: "imagens/laranja.jpg", preco: "6,00" },  
34    { id: 3, nome: "Uva", descricao: "Uva roxa, 500g", imageUrl: "imagens/uva.jpg", preco: "4,00" }  
35  ]  
36}
```

File Edit Selection View Go Run ... ← → ⌂ P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png

JS script.js

```
JS script.js > document.addEventListener("DOMContentLoaded") callback > categorias > "Bebidas"
31     "Bebidas": [
32         { id: 4, nome: "Coca Cola", descricao: "Refrigerante de cola, 2l", imageUrl: "imagens/coca.jpg", preco: "6,99" },
33         { id: 5, nome: "Suco de Laranja", descricao: "Suco natural, 1l", imageUrl: "imagens/suco.jpg", preco: "5,99" },
34         { id: 6, nome: "Água Mineral", descricao: "Água mineral sem gás, 500ml", imageUrl: "imagens/agua.jpg", preco: "1,99" }
35     ],
36     "Laticínios": [
37         { id: 7, nome: "Leite Integral", descricao: "Leite integral, 1l", imageUrl: "imagens/leite.jpg", preco: "3,59" },
38         { id: 8, nome: "Queijo", descricao: "Queijo", imageUrl: "imagens/queijo.jpg", preco: "50,90" },
39         { id: 9, nome: "Iogurte Natural", descricao: "Iogurte natural, 320g", imageUrl: "imagens/iogurte.jpg", preco: "2,99" }
40     ],
41     "Produtos de Limpeza": [
42         { id: 10, nome: "Detergente", descricao: "Detergente líquido, 500ml", imageUrl: "imagens/detergente.jpg", preco: "1,79" },
43         { id: 11, nome: "Desinfetante", descricao: "Desinfetante multiuso, 1l", imageUrl: "imagens/desinfetante.jpg", preco: "4,99" },
44         { id: 12, nome: "Água Sanitária", descricao: "Água sanitária, 2l", imageUrl: "imagens/agua_sanitaria.jpg", preco: "3,49" }
45     ],
46     "Snacks": [
47         { id: 13, nome: "Batata Chips", descricao: "Batata chips, sabor original, 100g", imageUrl: "imagens/batata_chips.jpg", preco: "6,99" },
48         { id: 14, nome: "Amendoim", descricao: "Amendoim torrado e salgado, 200g", imageUrl: "imagens/amendoim.jpg", preco: "5,49" },
49         { id: 15, nome: "Bolacha Recheada", descricao: "Bolacha recheada, sabor chocolate, 150g", imageUrl: "imagens/bolacha.jpg", preco: "2,50" }
50     ];
51 }
```

Ln 31, Col 13 Spaces: 8 UTF-8 CRLF {} JavaScript ⌂ Go Live



EXPLORER

P28 - PÁGINA SUPERMERCADO

> imagens

estilos.css

< index.html

p28_supermercado.png

JS script.js

JS script.js X

```
JS script.js > ⚡ document.addEventListener("DOMContentLoaded") callback
52     // Cada chave do objeto 'categorias' representa uma
53         // categoria de produto e contém um array de
54             // objetos que representam produtos individuais.
55
56     const secoesProdutos = document.getElementById('secoes-produtos');
57     // Obtém uma referência ao elemento HTML com o ID 'secoes-produtos'.
58     // Este elemento é usado para exibir as seções de produtos na página.
59
60     const campoPesquisa = document.getElementById('campo-pesquisa');
61     // Obtém uma referência ao campo de entrada de texto usado
62         // para pesquisar produtos, identificado pelo ID 'campo-pesquisa'.
63
64     const botaoPesquisar = document.getElementById('botao-pesquisar');
65     // Obtém uma referência ao botão de pesquisa, que é
66         // acionado para iniciar a busca de produtos
67             // baseada no texto inserido no 'campoPesquisa'.
68
69     const botaoVerCarrinho = document.getElementById('botao-ver-carrinho');
70     // Captura uma referência ao botão que, quando
71         // clicado, exibe o modal do carrinho de compras.
72
73     const contadorCarrinho = document.getElementById('contador-carrinho');
74     // Referência ao elemento que exibe o número de itens no
75         // carrinho, permitindo atualizações dinâmicas
76             // ao modificar o carrinho.
77
78     const listaItensCarrinho = document.getElementById('lista-itens-carrinho');
79     // Referência à lista no modal do carrinho onde os
80         // itens adicionados ao carrinho são exibidos.
81
82     const botoesCategoria = document.querySelectorAll('.categoria');
83     // Seleciona todos os elementos com a classe 'categoria', que
```



> OUTLINE



> TIMELINE



⊗ 0 △ 0 ⌂ 0

File Edit Selection View Go Run ... ← → ⌂ P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js

JS script.js

```
JS script.js > ⚡ document.addEventListener("DOMContentLoaded") callback
84     // são botões usados para filtrar produtos por
85     // categoria específica.
86
87     const botaoLimparFiltro = document.getElementById('limpar-filtro');
88     // Obtém uma referência ao botão que limpa todos os
89     // filtros ativos, mostrando todos os produtos disponíveis.
90
91     const modalProduto = new bootstrap.Modal(document.getElementById('modalProduto'));
92     // Cria uma instância do modal de Bootstrap para o
93     // produto, permitindo mostrar e esconder o
94     // modal programaticamente.
95
96     const imagemProduto = document.getElementById('imagem-produto');
97     // Referência ao elemento de imagem dentro do modal de
98     // produto, onde a imagem do produto selecionado é exibida.
99
100    const nomeProduto = document.getElementById('nome-produto');
101    // Referência ao elemento que exibe o nome do produto no
102    // modal de detalhes do produto.
103
104    const descricaoProduto = document.getElementById('descricao-produto');
105    // Obtém uma referência ao elemento que exibe a
106    // descrição do produto no modal de detalhes do produto.
107
108    const precoProduto = document.getElementById('preco-produto');
109    // Referência ao elemento que mostra o preço do
110    // produto no modal de detalhes do produto.
111
112    const quantidadeProduto = document.getElementById('quantidade-produto');
113    // Captura uma referência ao campo de entrada onde o
114    // usuário pode especificar a quantidade do
115    // produto que deseja adicionar ao carrinho.
```

Ln 84, Col 1 Spaces: 8 UTF-8 CRLF {} JavaScript ⌂ Go Live



EXPLORER

P28 - PÁGINA SUPERMERCADO

> imagens

estilos.css

< index.html

p28_supermercado.png

JS script.js

JS script.js



```
JS script.js > document.addEventListener("DOMContentLoaded") callback
116
117     const botaoAdicionarCarrinhoModal = document.getElementById('botao-adicionar-carrinho-modal');
118     // Referência ao botão no modal de detalhes do produto que,
119     // quando clicado, adiciona o produto ao carrinho com a
120     // quantidade especificada.
121
122
123
124     function atualizarExibicaoProdutos(categoriasFiltradas) {
125         // Define a função 'atualizarExibicaoProdutos', que é
126         // responsável por atualizar a seção de produtos no
127         // DOM com base nas categorias filtradas fornecidas.
128
129         secoesProdutos.innerHTML = '';
130         // Limpa todo o conteúdo dentro do elemento 'secoesProdutos'.
131         // Isso é necessário para remover a exibição anterior de
132         // produtos antes de adicionar novos produtos filtrados.
133
134         Object.keys(categoriasFiltradas).forEach(categoria => {
135             // Usa 'Object.keys' para obter um array de chaves do
136             // objeto 'categoriasFiltradas' (nomes das categorias).
137             // 'forEach' itera sobre cada chave (categoria) para
138             // processar os produtos dentro dessa categoria.
139
140             if (categoriasFiltradas[categoria].length > 0) {
141                 // Verifica se a lista de produtos na categoria atual
142                 // tem algum item. Se sim, procede à criação
143                 // de elementos para exibição.
144
145                 let secao = document.createElement('section');
146                 // Cria um novo elemento <section>, que será usado
147                 // para agrupar produtos de uma mesma categoria.
```



> OUTLINE

> TIMELINE



0 △ 0 ⌂ 0

File Edit Selection View Go Run ... ← → ⌂ P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png

JS script.js

JS script.js > document.addEventListener("DOMContentLoaded") callback > atualizarExibicaoProdutos > forEach() callback

```
148 let h2 = document.createElement('h2');
149 // Cria um elemento <h2>, que será usado para exibir o
150 // nome da categoria como um cabeçalho na seção.
151
152 h2.textContent = categoria;
153 // Define o texto do elemento <h2> para o nome da
154 // categoria, que serve como título para essa
155 // seção de produtos.
156
157
158 secao.appendChild(h2);
159 // Adiciona o elemento <h2> ao elemento <section>
160 // recém-criado, colocando o título na seção.
161
162 let linha = document.createElement('div');
163 // Cria um novo elemento <div> que será usado para
164 // organizar visualmente os produtos da
165 // categoria em uma linha.
166
167 linha.className = 'row';
168 // Atribui a classe 'row' ao elemento <div>. Esta é
169 // uma classe típica de frameworks CSS como
170 // Bootstrap, que usa o sistema de grid para
171 // alinhar elementos horizontalmente.
172
173 categoriasFiltradas[categoria].forEach(produto => {
174 // Itera sobre cada 'produto' no array de
175 // 'categoriasFiltradas' para a categoria atual.
176 // 'forEach' aplica uma função a cada item do array,
177 // permitindo manipular cada produto individualmente.
178
179 let coluna = document.createElement('div');
```

Ln 148, Col 1 Spaces: 8 UTF-8 CRLF {} JavaScript ⌂ Go Live

File Edit Selection View Go Run ... ← → 🔍 P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js

JS script.js X

JS script.js > ↗ document.addEventListener("DOMContentLoaded") callback > ↗ atualizarExibicaoProdutos > ↗ forEach() callback > ↗ forEach() callback

```
180 // Cria um novo elemento <div> que servirá como
181 // uma coluna no layout de grade (grid) do Bootstrap.
182
183 coluna.className = 'col-md-4';
184 // Atribui a classe 'col-md-4' ao elemento <div> recém-criado.
185 // Esta classe define que a coluna ocupa 4 de 12 colunas
186 // possíveis no grid para telas de tamanho
187 // médio (md), garantindo responsividade.
188
189 let cartao = document.createElement('div');
190 // Cria outro elemento <div> que será usado como
191 // um "cartão" para exibir as informações do produto.
192
193 cartao.className = 'card';
194 // Atribui a classe 'card' ao elemento <div>. Essa
195 // classe é parte do Bootstrap e é usada
196 // para aplicar um estilo de container com
197 // bordas e preenchimento.
198
199 // Configura o conteúdo interno do 'cartão' com as
200 // informações do produto.
201 cartao.innerHTML =
202 
204 <div class="card-body">
205   <h5 class="card-title">${produto.nome}</h5>
206   <p class="card-text">${produto.descricao}</p>
207   <p class="card-text"><strong>Preço: R$ ${produto.preco}</strong></p>
208   <button class="btn btn-primary adicionar-ao-carrinho" data-produto-id="${produto.id}">Adicionar ao Carrinho</button>
209 </div>;
```

Ln 180, Col 13 Spaces: 8 UTF-8 CRLF {} JavaScript ⚡ Go Live

File Edit Selection View Go Run ... ← → ⌂ P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js

JS script.js

```
JS script.js > document.addEventListener("DOMContentLoaded") callback > atualizarExibicaoProdutos > forEach() callback > forEach() callback
```

```
210 // Cria uma tag <img> com várias configurações:  
211 // 'src' define o caminho para a imagem do produto.  
212 // 'class="bg-imagem-topo"' aplica estilos para  
213 // garantir que a imagem se ajuste bem no topo do cartão.  
214 // 'alt' fornece texto alternativo para acessibilidade.  
215 // 'data-bs-toggle' e 'data-bs-target' são atributos  
216 // para controlar o modal do Bootstrap, permitindo  
217 // que ele seja aberto ao clicar na imagem.  
218 // 'data Produto-id' armazena o ID do produto, que  
219 // pode ser usado para identificar qual produto  
220 // deve ser mostrado no modal quando a imagem é clicada.  
221  
222 // Utiliza um <h5> para o nome do produto, com a  
223 // classe 'card-title' para estilo.  
224 // Exibe a descrição do produto em um <p> com a classe 'card-text'.  
225 // Mostra o preço do produto, destacando-o com a tag <strong>.  
226 // Cria um botão que permite adicionar o produto ao carrinho de compras.  
227 // 'btn btn-primary' são classes do Bootstrap para estilizar o botão.  
228 // 'data Produto-id' é usado para identificar qual  
229 // produto adicionar ao carrinho quando o botão é clicado.  
230  
231 coluna.appendChild(cartao);  
232 // Adiciona o 'cartão' à 'coluna'. Este cartão contém  
233 // todas as informações do produto e botões  
234 // de ação configurados acima.  
235  
236  
237 linha.appendChild(coluna);  
238 // Adiciona a 'coluna' à 'linha' existente, que foi  
239 // criada anteriormente para organizar os  
240 // produtos em uma grade horizontal.  
241
```

Ln 241, Col 17 Spaces: 8 UTF-8 CRLF {} JavaScript ⌂ Go Live

File Edit Selection View Go Run ... ← → P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png

JS script.js

```
JS script.js > document.addEventListener("DOMContentLoaded") callback > atualizarExibicaoProdutos > forEach() callback > forEach() callback
```

```
242     });
243
244     secao.appendChild(linha);
245     // Adiciona a 'linha' completa de produtos à 'secao'.
246
247     secoesProdutos.appendChild(secao);
248     // Finalmente, adiciona a 'secao' ao elemento
249     // principal 'secoesProdutos' que contém
250     // todas as seções de produtos.
251
252 }
253 );
254
255
256 // Adiciona os eventos de clique nos botões "Adicionar ao Carrinho"
257 document.querySelectorAll('.adicionar-ao-carrinho').forEach(botao => {
258
259     // Utiliza 'document.querySelectorAll' para selecionar
260     // todos os elementos no documento com a
261     // classe 'adicionar-ao-carrinho'.
262     // 'forEach' é aplicado a cada elemento retornado
263     // pela seleção, e para cada botão encontrado, a
264     // função fornecida é executada.
265     // 'botao' é a variável que representa cada elemento
266     // iterado, que neste contexto, é cada botão
267     // de adicionar ao carrinho.
268
269     botao.addEventListener('click', function() {
270         // Adiciona um ouvinte de evento de clique ao 'botao'.
271         // Quando o botão é clicado, a função
272         // anônima (função sem nome) especificada é chamada.
273         // 'addEventListener' é um método que registra uma
```

Ln 242, Col 9 Spaces: 8 UTF-8 CRLF {} JavaScript Go Live

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js

JS script.js

```
JS script.js > document.addEventListener("DOMContentLoaded") callback > atualizarExibicaoProdutos > forEach() callback > botao.addEventListener("click") callback
```

274 // função para ser chamada sempre que o
275 // evento especificado (neste caso, 'click')
276 // acontece no elemento.

277

278 const idProduto = parseInt(this.getAttribute('data-produto-id'));
279 // 'this' dentro do manipulador de evento se
280 // refere ao elemento ao qual o ouvinte de
281 // evento foi adicionado, ou seja, o botão
282 // que foi clicado.
283 // 'getAttribute' é um método que obtém o valor de
284 // um atributo do elemento; 'data-produto-id' é
285 // um atributo personalizado definido no HTML que
286 // armazena o ID do produto.
287 // 'parseInt' é uma função que converte seu argumento de
288 // string para um inteiro. Isso é usado aqui para
289 // transformar o ID do produto de uma string
290 // para um número, pois os IDs geralmente são
291 // manipulados como números.

292

293 adicionarAoCarrinho(idProduto);
294 // Chama a função 'adicionarAoCarrinho', passando
295 // o 'idProduto' como argumento.
296 // Esta função é responsável por adicionar o produto
297 // especificado ao carrinho de compras, utilizando o
298 // ID para identificar qual produto adicionar.

299

300);
301);

302

303

304 // Adiciona os eventos de clique nas imagens dos
305 // produtos para abrir o modal de detalhes

The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Toolbar:** Back, Forward, Home, P28 - Página Supermercado, Minimize, Maximize, Close.
- Left Sidebar (EXPLORER):** Shows a tree view of files under "P28 - PÁGINA SUPERMERCADO":
 - imagens
 - # estilos.css
 - index.html
 - p28_supermercado.png
 - script.js (selected)
- Code Editor:** The active tab is "JS script.js". The code is as follows:

```
JS script.js > document.addEventListener("DOMContentLoaded") callback > atualizarExibicaoProdutos
306     document.querySelectorAll('.bg-imagem-topo').forEach(imagem => {
307         // Utiliza 'document.querySelectorAll' para selecionar
308         // todas as imagens que têm a classe 'bg-imagem-topo'.
309         // O método 'forEach' itera sobre cada imagem encontrada,
310         // executando a função fornecida para cada uma.
311
312         imagem.addEventListener('click', function() {
313             // Adiciona um ouvinte de evento de clique a cada
314             // imagem. Quando a imagem é clicada, a
315             // função anônima é executada.
316             // 'addEventListener' é um método que vincula uma
317             // função a ser chamada sempre que o evento
318             // especificado (neste caso, 'click') ocorre no elemento.
319
320             const idProduto = parseInt(this.getAttribute('data-produto-id'));
321             // 'this' refere-se ao elemento imagem que foi clicado.
322             // 'getAttribute' obtém o valor do atributo 'data-produto-id'
323             // da imagem, que armazena o ID do produto associado.
324             // 'parseInt' converte a string do ID em um número
325             // inteiro, que é necessário para processamento posterior.
326
327             const produto = obterProdutoPorId(idProduto);
328             // Chama a função 'obterProdutoPorId', passando o 'idProduto'
329             // como argumento para buscar os detalhes
330             // completos do produto no banco de dados ou na
331             // estrutura de dados local.
332
333             if (produto) {
334                 // Verifica se um produto foi efetivamente retornado
335                 // pela função 'obterProdutoPorId'.
336
337                 imagemProduto.src = produto.imageUrl;
```
- Bottom Status Bar:** Ln 306, Col 3, Spaces: 8, UTF-8, CRLF, {}, JavaScript, Go Live.

File Edit Selection View Go Run ... ← → 🔍 P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png

JS script.js

JS script.js > document.addEventListener("DOMContentLoaded") callback > atualizarExibicaoProdutos > forEach() callback > imagem.addEventListener("click")

```
338     // Atualiza a propriedade 'src' do elemento 'imagemProduto'  
339         // para mostrar a imagem do produto selecionado no modal.  
340  
341     nomeProduto.textContent = produto.nome;  
342     // Define o texto do elemento 'nomeProduto' para o  
343         // nome do produto, atualizando o conteúdo no modal.  
344  
345     descricaoProduto.textContent = produto.descricao;  
346     // Atualiza a descrição do produto no  
347         // elemento 'descricaoProduto' dentro do modal.  
348  
349     precoProduto.textContent = produto.preco;  
350     // Exibe o preço do produto no elemento 'precoProduto'.  
351  
352     botaoAdicionarCarrinhoModal.setAttribute('data-produto-id', produto.id);  
353     // Define o atributo 'data-produto-id' do botão  
354         // 'botaoAdicionarCarrinhoModal' para o ID do produto,  
355         // permitindo que o botão saiba qual produto  
356         // adicionar ao carrinho se clicado.  
357  
358     quantidadeProduto.value = 1; // Resetar a quantidade para 1  
359     // Reseta o valor do campo de quantidade no  
360         // modal para 1 cada vez que o modal é  
361         // aberto, garantindo que o usuário comece  
362         // com uma quantidade padrão.  
363  
364     modalProduto.show();  
365     // Chama o método 'show' do 'modalProduto' para  
366         // exibir o modal, tornando-o visível na  
367         // tela com os detalhes do produto atualizados.  
368  
369 }
```

Ln 338, Col 8 Spaces: 8 UTF-8 CRLF {} JavaScript ⚡ Go Live



EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png

JS script.js

JS script.js

```
JS script.js > ↗ document.addEventListener("DOMContentLoaded") callback > ↗ atualizarExibicaoProdutos > ↗ forEach() callback > ↗ imagem.addEventListener('click')
```

```
370     |    });
371     });
372
373 }
374
375
376
377 function adicionarAoCarrinho(idProduto, quantidade = 1) {
378     // Define a função 'adicionarAoCarrinho', que adiciona um
379     // produto ao carrinho ou incrementa sua
380     // quantidade se já estiver presente.
381     // 'idProduto' é o identificador único do
382     // produto a ser adicionado.
383     // 'quantidade' é o número de unidades do produto a
384     // ser adicionado. Se não especificado, o padrão é 1.
385
386     const produto = obterProdutoPorId(idProduto);
387     // Chama a função 'obterProdutoPorId' para buscar o
388     // produto pelo seu 'idProduto' dentro da
389     // estrutura de dados que representa o
390     // estoque ou catálogo.
391     // 'produto' será o objeto representando o produto se
392     // encontrado, ou 'undefined' se não houver
393     // produto com esse ID.
394
395     if (!produto) return;
396     // Verifica se o produto foi encontrado.
397     // Se 'produto' for 'undefined' (não encontrado),
398     // a função termina prematuramente.
399     // 'return' sem um valor encerra a execução da
400     // função sem fazer mais nada.
```

> OUTLINE

> TIMELINE

The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Toolbar:** Back, Forward, Home, P28 - Página Supermercado, Minimize, Maximize, Close.
- Sidebar (EXPLORER):** Shows a project structure under "P28 - PÁGINA SUPERMERCADO":
 - imágenes
 - # estilos.css
 - index.html
 - p28_supermercado.png
 - JS script.js (selected)
- Code Editor:** The active file is "script.js". The code is a function named "adicionarAoCarrinho" which adds items to a shopping cart array. It uses a callback function for the DOMContentLoaded event.

```
JS script.js > document.addEventListener("DOMContentLoaded") callback > adicionarAoCarrinho
402     const produtoExistente = carrinho.find(item => item.id === produto.id);
403     // Busca no array 'carrinho' para ver se o
404     //     // produto já existe nele.
405     // 'find()' retorna o primeiro elemento que
406     //     // satisfaz a condição especificada na função de
407     //     // callback, ou 'undefined' se nenhum elemento a satisfaz.
408
409     if (produtoExistente) {
410
411         produtoExistente.quantidade += quantidade;
412         // Se o produto já existe no carrinho ('produtoExistente' não
413         //     // é 'undefined'), incrementa a 'quantidade'
414         // existente do produto.
415
416     } else {
417
418         carrinho.push({ ...produto, quantidade: quantidade });
419         // Se o produto não está no carrinho, adiciona-o com a
420         //     // propriedade 'quantidade' especificada.
421         // '{ ...produto, quantidade: quantidade }' cria um novo
422         //     // objeto copiando todas as propriedades de 'produto' e
423         //     // adicionando/atualizando a propriedade 'quantidade'.
424
425     }
426
427     atualizarContadorCarrinho();
428     // Chama a função 'atualizarContadorCarrinho' para
429     //     // atualizar a exibição do contador de itens no
430     //     // carrinho e salvar o estado atualizado no localStorage.
431
432 }
433
```
- Bottom Status Bar:** Line 402, Column 8, Spaces: 8, UTF-8, CRLF, {}, JavaScript, Go Live.

File Edit Selection View Go Run ... ← → ⌂ P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js

JS script.js

JS script.js > ⌂ document.addEventListener("DOMContentLoaded") callback

```
434
435     function obterProdutoPorId(id) {
436         // Define a função 'obterProdutoPorId', que busca
437         // um produto pelo seu ID dentro das
438         // categorias de produtos disponíveis.
439
440         for (let categoria in categorias) {
441             // Itera sobre cada categoria no objeto 'categorias'. 'categoria'
442             // refere-se à chave do objeto (nome da categoria).
443
444             const produto = categorias[categoria].find(item => item.id === id);
445             // Dentro de cada categoria, usa 'find()' para
446             // buscar o primeiro produto que tenha um 'id'
447             // que corresponda ao 'id' fornecido.
448             // 'find()' retorna o objeto do produto se
449             // encontrado, ou 'undefined' se nenhum
450             // produto corresponder.
451
452             if (produto) return produto;
453             // Se um produto é encontrado (ou seja, 'produto'
454             // não é 'undefined'), retorna esse produto
455             // imediatamente, terminando a função.
456
457         }
458
459         return null;
460         // Após verificar todas as categorias, se nenhum
461         // produto for encontrado com o 'id'
462         // especificado, retorna 'null'.
463         // 'null' é usado aqui para indicar explicitamente
464         // que nenhum produto foi encontrado.
465     }
```

Ln 434, Col 1 Spaces: 8 UTF-8 CRLF {} JavaScript Go Live

File Edit Selection View Go Run ... ← → P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js

JS script.js

```
JS script.js > document.addEventListener("DOMContentLoaded") callback > obterProdutoPorId
466    }
467
468
469    function exibirCarrinho() {
470        // Define a função 'exibirCarrinho', responsável por
471        // atualizar e exibir o conteúdo do carrinho
472        // de compras no modal correspondente.
473
474        listaItensCarrinho.innerHTML = ''; // Limpa a lista
475        // antes de adicionar os itens
476        // Define o conteúdo interno de 'listaItensCarrinho' (um
477        // elemento <ul> ou <ol>) como vazio, removendo
478        // quaisquer elementos filho que existiam antes, para
479        // preparar para uma nova listagem de itens.
480
481        if (carrinho.length === 0) {
482            // Verifica se o array 'carrinho' está vazio.
483
484            let mensagemVazia = document.createElement('li');
485            // Cria um novo elemento <li> que será usado para
486            // mostrar uma mensagem quando o carrinho estiver vazio.
487
488            mensagemVazia.textContent = 'Seu carrinho está vazio.';
489            // Define o texto do elemento <li> para informar ao
490            // usuário que não há itens no carrinho.
491
492            mensagemVazia.className = 'list-group-item';
493            // Atribui a classe 'list-group-item' ao elemento <li> para
494            // estilização consistente com o framework
495            // Bootstrap ou CSS personalizado.
496
497            listaItensCarrinho.appendChild(mensagemVazia);
```

Ln 466, Col 1 Spaces: 8 UTF-8 CRLF {} JavaScript Go Live

File Edit Selection View Go Run ... ← → 🔍 P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js

JS script.js

```
JS script.js > ⚡ document.addEventListener("DOMContentLoaded") callback > ⚡ exibirCarrinho
498         // Adiciona o elemento <li> ao elemento pai 'listaItensCarrinho',
499         |   // efetivamente exibindo a mensagem de carrinho vazio.
500
501     } else {
502         // Caso haja itens no carrinho, procede para listá-los.
503
504         carrinho.forEach(item => {
505             // Itera sobre cada 'item' no array 'carrinho'.
506
507             let itemLista = document.createElement('li');
508             // Cria um novo elemento <li> para cada item do carrinho.
509
510             itemLista.className = 'list-group-item d-flex justify-content-between align-items-center';
511             // Atribui classes para estilização e layout
512             |   // flexível, usando Bootstrap para alinhar os
513             |   // conteúdos de forma responsiva e esteticamente agradável.
514
515             itemLista.innerHTML =
516                 <div class="item-carrinho">
517                     
518                     <div class="detalhes-item-carrinho">
519                         <span>${item.nome}</span>
520                         <span>R$ ${item.preco}</span>
521                         <span>Quantidade: ${item.quantidade}</span>
522                     </div>
523                 </div>
524                 <button class="btn btn-danger btn-sm remover-do-carrinho" data-produto-id="${item.id}">Remover</
525             button>
526         ;
527         // Configura o HTML interno do <li>, incluindo uma
528             |   // divisão com a imagem do produto, detalhes
529             |   // como nome, preço, e quantidade,
```

Ln 498, Col 4 Spaces: 8 UTF-8 CRLF {} JavaScript ⚡ Go Live

File Edit Selection View Go Run ... ← → ⌂ P28 - Página Supermercado

EXPLORER

P28 - PÁGINA SUPERMERCADO

- > imagens
- # estilos.css
- index.html
- p28_supermercado.png
- JS script.js

JS script.js

```
JS script.js > document.addEventListener("DOMContentLoaded") callback > exibirCarrinho > carrinho.forEach() callback
529         // e um botão para remover o item do carrinho, utilizando
530             // atributos 'data-' para identificar qual item remover.
531
532     listaItensCarrinho.appendChild(itemLista);
533     // Adiciona o elemento <li> configurado à lista 'listaItensCarrinho',
534         // exibindo o item no carrinho.
535
536 });
537
538 let itemTotal = document.createElement('li');
539 // Cria outro elemento <li> para exibir o total do carrinho.
540
541 itemTotal.className = 'list-group-item list-group-item-dark';
542 // Atribui classes para estilização, usando uma
543     // classe Bootstrap para diferenciar visualmente o
544         // total dos itens individuais.
545
546 let total = carrinho.reduce((acc, item) => acc + (parseFloat(item.preco.replace(',', '.')) * item.
547     quantidade), 0);
548 // Calcula o total do carrinho. Converte o preço de
549     // string para float e soma os produtos das
550         // quantidades pelos preços.
551
552 itemTotal.textContent = `Total: R$ ${total.toFixed(2)}`;
553 // Define o texto do elemento <li> do total para mostrar o
554     // valor total calculado, formatado para duas casas decimais.
555
556 listaItensCarrinho.appendChild(itemTotal);
557 // Adiciona o elemento <li> do total à lista 'listaItensCarrinho',
558     // exibindo o total no carrinho.
559 }
```

Ln 529, Col 6 Spaces: 8 UTF-8 CRLF {} JavaScript ⌂ Go Live

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Title Bar:** P28 - Página Supermercado
- Left Sidebar (Explorer):**
 - P28 - PÁGINA SUPERMERCADO
 - imagens
 - # estilos.css
 - index.html
 - p28_supermercado.png
 - JS script.js
- Central Area (Code Editor):**

JS script.js > document.addEventListener("DOMContentLoaded") callback > exibirCarrinho

```
560
561
562    // Adiciona os eventos de clique nos botões "Remover"
563    document.querySelectorAll('.remover-do-carrinho').forEach(botao => {
564        // Usa 'document.querySelectorAll' para selecionar todos os
565        // elementos com a classe 'remover-do-carrinho'.
566        // 'forEach' itera sobre cada botão encontrado, executando
567        // a função fornecida para cada um.
568
569        botao.addEventListener('click', function() {
570            // Adiciona um ouvinte de evento de clique ao botão. Quando
571            // clicado, a função anônima será executada.
572
573            const idProduto = parseInt(this.getAttribute('data-produto-id'));
574            // Dentro do manipulador de clique, 'this' refere-se
575            // ao botão que foi clicado.
576            // 'getAttribute' obtém o valor do atributo 'data-produto-id',
577            // que é uma string contendo o ID do produto.
578            // 'parseInt' converte essa string para um número inteiro,
579            // que é o ID do produto a ser removido.
580
581            removerDoCarrinho(idProduto);
582            // Chama a função 'removerDoCarrinho' com o 'idProduto'
583            // como argumento, que remove o produto
584            // especificado do carrinho.
585
586        });
587    });
588
589    let modalCarrinho = new bootstrap.Modal(document.getElementById('modalCarrinho'), {});
590    // Cria uma nova instância do modal do Bootstrap para o
591    // elemento com ID 'modalCarrinho'.
```
- Bottom Status Bar:** Ln 560, Col 5, Spaces: 8, UTF-8, CRLF, {}, JavaScript, Go Live

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top bar includes the standard VS Code icons and menu items: File, Edit, Selection, View, Go, Run, and others. A search bar at the top right contains the text "P28 - Página Supermercado". The left sidebar has several icons: Explorer, Search, Open, Issues, and Outline/Timeline. The Explorer section shows a folder named "P28 - PÁGINA SUPERMERCADO" containing files: "imagens", "# estilos.css", "index.html", "p28_supermercado.png", and "script.js". The "script.js" file is currently selected and open in the main editor area. The editor shows the following code:

```
JS script.js > document.addEventListener("DOMContentLoaded") callback > removerDoCarrinho
592     // A instância é armazenada na variável 'modalCarrinho'. Um
593         // objeto de opções vazio é passado como segundo argumento.
594
595     modalCarrinho.show();
596     // Chama o método 'show' na instância 'modalCarrinho' para
597         // exibir o modal. Isso faz o modal se tornar visível
598         // na interface do usuário.
599
600     modalCarrinho._element.addEventListener('hidden.bs.modal', function() {
601         // Adiciona um ouvinte de evento ao elemento do modal
602             // que detecta quando o modal é fechado.
603             // 'hidden.bs.modal' é um evento específico do Bootstrap
604                 // que é disparado após o modal ser completamente escondido.
605
606         window.location.href = "index.html";
607         // Redireciona o navegador para a página 'index.html'.
608         // Esta linha é executada quando o evento 'hidden.bs.modal' é
609             // disparado, indicando que o modal foi fechado.
610
611     });
612
613 }
614
615
616 function removerDoCarrinho(idProduto) {
617     // Define a função 'removerDoCarrinho', que remove um
618         // produto do carrinho de compras com base no seu ID.
619
620     carrinho = carrinho.filter(item => item.id !== idProduto);
621     // Atualiza o array 'carrinho' para incluir apenas os
```

The status bar at the bottom shows: ⌂ 0 △ 0 ⌂ 0 Ln 621, Col 4 Spaces: 8 UTF-8 CRLF {} JavaScript ⌂ Go Live ⌂

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top bar includes the standard VS Code menu items: File, Edit, Selection, View, Go, Run, and a three-dot ellipsis. To the right of the menu is a search bar containing the text "P28 - Página Supermercado". On the far right are window control buttons (minimize, maximize, close) and a set of icons for document preview, file operations, and other settings.

The left sidebar contains several icons: a clipboard (EXPLORER), a magnifying glass (SEARCH), a gear (OPTIONS), a person (USERS), and two square icons (OUTLINE and TIMELINE). The "OUTLINE" icon is currently selected.

The main workspace shows a file named "script.js" with the extension ".js" highlighted. The file content is as follows:

```
document.addEventListener("DOMContentLoaded") callback > atualizarContadorCarrinho
652     // incluir a palavra "itens" após o número.
653
654     localStorage.setItem('carrinho', JSON.stringify(carrinho));
655     // Salva o estado atual do carrinho no armazenamento
656     // local do navegador para persistência de dados.
657     // 'JSON.stringify(carrinho)' converte o array 'carrinho' em
658     // uma string JSON para que possa ser armazenado
659     // corretamente, pois o armazenamento local só
660     // suporta strings.
661 }
663
664
665 botaoAdicionarCarrinhoModal.addEventListener('click', function() {
666     // Adiciona um ouvinte de evento de clique ao
667     // botão 'botaoAdicionarCarrinhoModal'.
668     // Quando o botão é clicado, a função anônima
669     // especificada é executada.
670     // 'addEventListener' é um método usado para registrar um
671     // manipulador de evento, neste caso, para o
672     // evento de clique.
673
674     const idProduto = parseInt(this.getAttribute('data-produto-id'));
675     // Dentro da função, 'this' refere-se ao botão
676     // 'botaoAdicionarCarrinhoModal' que foi clicado.
677     // 'getAttribute' é usado para recuperar o valor do
678     // atributo 'data-produto-id', que armazena o
679     // ID do produto associado ao botão.
680     // 'parseInt' converte o valor do ID, que é uma
681     // string, em um número inteiro. Isso é necessário
```

The code is written in Portuguese and handles the addition of items to a shopping cart. It uses event listeners and the `localStorage` API to persist the cart state across page reloads. It also demonstrates how to get the value of an attribute from the current element using `this.getAttribute` and convert it to an integer using `parseInt`.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top bar includes standard menu items like File, Edit, Selection, View, Go, Run, and a search bar labeled "P28 - Página Supermercado". The left sidebar has icons for Explorer, Search, Problems, and others, with "OUTLINE" and "TIMELINE" also visible. The main editor area displays a JavaScript file named "script.js". The code is annotated with comments explaining its logic:

```
// para a manipulação correta do ID no JavaScript.  
const quantidade = parseInt(quantidadeProduto.value);  
// 'quantidadeProduto' é um elemento de entrada no  
// modal onde o usuário pode especificar a  
// quantidade do produto que deseja adicionar ao carrinho.  
// 'quantidadeProduto.value' acessa o valor atual  
// dessa entrada, que é uma string.  
// 'parseInt' também é usado aqui para converter essa  
// string para um número inteiro, representando a  
// quantidade do produto a ser adicionada ao carrinho.  
  
adicionarAoCarrinho(idProduto, quantidade);  
// Chama a função 'adicionarAoCarrinho',  
// passando 'idProduto' e 'quantidade' como argumentos.  
// Esta função é responsável por adicionar o produto  
// especificado, na quantidade especificada,  
// ao carrinho de compras.  
  
modalProduto.hide();  
// Após adicionar o item ao carrinho, este comando  
// oculta o modal de detalhes do produto.  
// 'hide' é um método do objeto modal do Bootstrap que  
// fecha o modal, removendo-o da visualização do usuário.  
});  
  
botaoPesquisar.addEventListener('click', function() {  
// Adiciona um ouvinte de evento de clique ao
```

The code handles user input for product quantity, calls an "adicionarAoCarrinho" function with the product ID and quantity, and then hides the product details modal using the Bootstrap "hide" method.

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P28 - Página Supermercado
- Icons:** Explorer, Search, Open, Save, Help, Outline, Timeline.
- Left Sidebar (Explorer):** P28 - PÁGINA SUPERMERCADO (expanded) containing: imagens, # estilos.css, index.html, p28_supermercado.png, script.js.
- Active Editor Tab:** script.js
- Code Content:** A snippet of JavaScript code for a search functionality. The code uses event listeners, string manipulation (toLowerCase), object iteration (Object.keys, forEach), and array filtering (filter).

```
document.addEventListener("DOMContentLoaded") callback > botaoPesquisar.addEventListener('click') callback
    // botão 'botaoPesquisar'.
    // Quando o botão é clicado, a função anônima
    // especificada é executada.
    // 'addEventListener' é um método que registra um
    // manipulador de eventos, neste caso, para o
    // evento de clique.

const termoPesquisa = campoPesquisa.value.toLowerCase();
// Obtém o valor atual do campo de pesquisa 'campoPesquisa',
// que é onde os usuários inserem o texto que
// desejam pesquisar.
// 'toLowerCase' é usado para converter o texto de
// pesquisa para letras minúsculas, garantindo que a
// pesquisa não seja sensível a maiúsculas ou minúsculas.

const categoriasFiltradas = {};
// Inicializa um objeto vazio 'categoriasFiltradas'. Este
// objeto será usado para armazenar as categorias e os
// produtos filtrados que correspondem ao termo de pesquisa.

Object.keys(categorias).forEach(categoria => {
    // 'Object.keys(categorias)' obtém todas as chaves do
    // objeto 'categorias' (que são os nomes das
    // categorias de produtos).
    // 'forEach' é usado para iterar sobre cada categoria.

    categoriasFiltradas[categoria] = categorias[categoria].filter(produto => {
        // Para cada categoria, filtra os produtos que
        // incluem o termo de pesquisa no nome.
        // 'filter' é um método de array que cria um novo
```

- Bottom Status Bar:** 0 0 0 0, Ln 712, Col 5, Spaces: 8, UTF-8, CRLF, {} JavaScript, Go Live, ...

The screenshot shows a Microsoft Edge browser window with developer tools open. The title bar says 'P28 - Página Supermercado'. The left sidebar has icons for Explorer, Search, Open, and Outline/Timeline. The Explorer section shows files: 'PÁGINA SUPERMERCADO' (containing 'imagens', '# estilos.css', 'index.html', and 'p28_supermercado.png'), and 'script.js' which is selected. The main code editor shows the following JavaScript code:

```
document.addEventListener("DOMContentLoaded") callback > botaoPesquisar.addEventListener('click') callback > forEach() callback > filter() callback

742         // array com todos os elementos que passam no
743         // teste implementado pela função fornecida.
744
745     return produto.nome.toLowerCase().includes(termoPesquisa);
746         // 'toLowerCase' é usado para garantir que a
747             // comparação do nome do produto com o
748                 // termo de pesquisa seja feita de maneira
749                     // não sensível a maiúsculas/minúsculas.
750
751         // 'includes' verifica se o nome do produto contém o
752             // termo de pesquisa fornecido.
753
754     });
755
756     atualizarExibicaoProdutos(categoriasFiltradas);
757         // Chama a função 'atualizarExibicaoProdutos' com o
758             // objeto 'categoriasFiltradas'.
759
760             // Esta função é responsável por atualizar a interface do
761                 // usuário para mostrar apenas os produtos que
762                     // correspondem ao termo de pesquisa.
763
764
765
766     botoesCategoria.forEach(botao => {
767         // 'botoesCategoria' é uma coleção de elementos de botão,
768             // cada um representando uma categoria de produto.
769
770             // 'forEach' itera sobre cada botão na coleção, executando a
771                 // função fornecida para cada um.
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top bar includes icons for File, Edit, Selection, View, Go, Run, and a search bar labeled "P28 - Página Supermercado". The left sidebar has icons for Explorer, Search, Open, and Outline/Timeline. The Explorer view shows a folder named "P28 - PÁGINA SUPERMERCADO" containing files: "imagens", "# estilos.css", "index.html", "p28_supermercado.png", and "script.js". The main editor area displays the "script.js" file with the following code:

```
JS script.js  X JS script.js > document.addEventListener("DOMContentLoaded") callback > botoesCategoria.forEach() callback

772     botao.addEventListener('click', function() {
773         // Adiciona um ouvinte de evento de clique a
774         // cada botão de categoria.
775         // Quando um botão é clicado, a função anônima
776         // especificada é chamada.
777         // 'addEventListener' é um método que registra uma
778         // função a ser chamada sempre que o evento
779         // especificado (neste caso, 'click') ocorre no elemento.
780
781     const categoria = this.getAttribute('data-categoria');
782     // 'this' refere-se ao botão de categoria que foi clicado.
783     // 'getAttribute' é usado para recuperar o valor do
784     // atributo 'data-categoria' do botão.
785     // Este atributo 'data-categoria' contém o nome
786     // da categoria que o botão representa.
787
788     const categoriasFiltradas = {};
789     // Inicializa um objeto vazio 'categoriasFiltradas'.
790     // Este objeto será usado para armazenar a
791     // categoria específica e seus produtos correspondentes.
792
793     categoriasFiltradas[categoria] = categorias[categoria];
794     // Atribui ao objeto 'categoriasFiltradas' uma
795     // chave que é o nome da categoria com o valor
796     // sendo a lista de produtos dessa categoria.
797     // 'categorias[categoria]' acessa a lista de produtos para a
798     // categoria específica do objeto 'categorias'.
799
800     atualizarExibicaoProdutos(categoriasFiltradas);
801     // Chama a função 'atualizarExibicaoProdutos', passando o
```

A screenshot of a Microsoft Edge browser window displaying a developer tools interface. The title bar shows "P28 - Página Supermercado". The left sidebar contains icons for Explorer, Search, Open, and Outline/Timeline. The main area shows the "script.js" file with code related to a supermarket page.

File Edit Selection View Go Run ... ← → 🔍 P28 - Página Supermercado

EXPLORER ... JS script.js X

P28 - PÁGINA SUPERMERCADO

> imagens
estilos.css
↳ index.html
🖼️ p28_supermercado.png
JS script.js

JS script.js > document.addEventListener("DOMContentLoaded") callback > botoesCategoria.forEach() callback > botao.addEventListener('click') callback

```
802         // objeto 'categoriasFiltradas' como argumento.
803         // Esta função é responsável por atualizar a interface do
804         // usuário para mostrar apenas os produtos da
805         // categoria selecionada.
806     });
807 );
808 );
809
810
811 botaoLimparFiltro.addEventListener('click', function() {
812     // Adiciona um ouvinte de evento de clique ao botão 'botaoLimparFiltro'.
813     // Este botão é responsável por remover todos os
814     // filtros atuais aplicados à visualização de produtos.
815     // Quando o botão é clicado, a função anônima
816     // especificada é executada.
817
818     atualizarExibicaoProdutos(categorias);
819     // Chama a função 'atualizarExibicaoProdutos' passando o
820     // objeto 'categorias' inteiro.
821     // Isso faz com que a exibição de produtos na
822     // interface do usuário seja atualizada para
823     // mostrar todos os produtos disponíveis,
824     // independentemente da categoria.
825
826 );
827
828     // Inicializa a lista com todos os produtos
829     atualizarExibicaoProdutos(categorias);
830     // Esta linha é executada assim que o script é carregado,
831     // garantindo que todos os produtos sejam exibidos inicialmente.
```

Ln 802, Col 3 Spaces: 8 UTF-8 CRLF {} JavaScript ⚡ Go Live



```
JS script.js > document.addEventListener("DOMContentLoaded") callback
832     // Chama a função 'atualizarExibicaoProdutos' com o objeto
833         // completo de 'categorias', o que resulta na criação
834             // inicial da exibição de todos os produtos disponíveis.
835
836     botaoVerCarrinho.addEventListener('click', exibirCarrinho);
837         // Adiciona um ouvinte de evento de clique ao botão 'botaoVerCarrinho'.
838         // 'exibirCarrinho' é uma função que, quando chamada, mostra o
839             // modal ou a interface com os itens atualmente no
840                 // carrinho de compras.
841         // Este ouvinte vincula diretamente a função 'exibirCarrinho' ao
842             // evento de clique, facilitando a visualização do
843                 // carrinho pelo usuário a qualquer momento.
844
845     // Atualiza a contagem ao carregar a página
846     atualizarContadorCarrinho();
847         // Esta chamada de função é executada assim que o script
848             // carrega, atualizando o contador de itens no carrinho.
849         // 'atualizarContadorCarrinho' verifica quantos itens
850             // estão atualmente no carrinho (usando dados
851                 // salvos no localStorage) e atualiza a
852                     // interface do usuário para refletir essa quantidade.
853
854
855 });


```

UC: Projetos

DIAS: 25/10/2024

Assunto:

Small Projects

P29 – Jogo da Velha

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Help, User, Outline, Timeline.
- Left Sidebar (EXPLORER):** P29 - JOGO DA VELHA (expanded) containing # estilos.css, <jogo.html>, p29_jogo_da_velha.png, and script.js.
- Active Tab:** Welcome > jogo.html
- Code Editor Content:** The code for the jogo.html file is displayed, starting with the DOCTYPE declaration and the HTML root element. The code is annotated with explanatory comments in green text.
- Bottom Status Bar:** Includes icons for file operations (Save, Undo, Redo), status (Ln 1, Col 1), and file format (Spaces: 8, UTF-8, CRLF, HTML).

```
<!DOCTYPE html>
<!-- A declaração DOCTYPE é crucial para definir o padrão de HTML que o navegador deve usar para interpretar o documento. '!DOCTYPE html' especifica que este documento segue o padrão HTML5, que é a versão mais recente e recomendada do HTML. Essa linha deve ser a primeira no seu documento HTML, para garantir que o navegador processe tudo corretamente. --&gt;
&lt;html lang="pt-br"&gt;
<!-- A tag &lt;html&gt; é o elemento raiz de todo documento HTML. Ela deve englobar todos os outros elementos HTML (exceto o DOCTYPE). O atributo 'lang="pt-br"' especifica que o idioma principal do documento é o português do Brasil. Isso ajuda motores de busca e navegadores a tratar o conteúdo do documento de forma adequada, além de ser importante para acessibilidade. --&gt;
&lt;head&gt;
<!-- A tag &lt;head&gt; contém metadados, links para folhas de estilo, scripts e outras informações que não são diretamente visíveis na página. Tudo dentro de &lt;head&gt; é essencial para configurar a página, mas não constitui o conteúdo que os usuários veem no navegador. --&gt;
&lt;meta charset="UTF-8"&gt;</pre>
```

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Find, Outline, Timeline.
- Left Sidebar:** P29 - JOGO DA VELHA (expanded) containing files: # estilos.css, < jgo.html, p29_jogo_da_velha.png, JS script.js.
- Active Tab:** Welcome > jogo.html
- Code Editor:** The file 'jogo.html' contains the following code with annotations:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <!-- A tag <meta> com o atributo 'charset="UTF-8"' define a codificação de caracteres para UTF-8, que suporta quase todos os caracteres de qualquer idioma. Isso é crucial para garantir que qualquer texto dentro da página seja exibido corretamente. -->

        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <!-- Esta tag <meta> configura a viewport (área de visualização) da página para dispositivos móveis.
            - 'width=device-width' faz com que a largura da página corresponda à largura do dispositivo.
            - 'initial-scale=1.0' define o nível inicial de zoom quando a página é carregada pela primeira vez.
            Essas configurações são importantes para garantir que a página seja responsiva e bem visualizada em diferentes dispositivos. -->

        <title>Jogo da Velha</title>
        <!-- A tag <title> define o título da página, que é exibido na aba do navegador. Este título também é usado por motores de busca, sendo importante para SEO. -->

        <link rel="stylesheet" href="estilos.css" />
        <!-- A tag <link> com o atributo 'rel="stylesheet"' é usada para incluir uma folha de estilo CSS externa no documento. O atributo 'href' especifica o caminho para o arquivo CSS ('estilos.css'), que contém os estilos que serão aplicados ao documento.
            Incluir CSS desta maneira é fundamental para controlar a aparência da página. -->
    </head>
    <body>
        <h1>Jogo da Velha</h1>
        <div id="board">
            <table border="1">
                <tr><td></td><td></td><td></td></tr>
                <tr><td></td><td></td><td></td></tr>
                <tr><td></td><td></td><td></td></tr>
            </table>
        </div>
        <div id="status">Jogador X: <span></span> | Jogador O: <span></span></div>
        <div id="message"></div>
    </body>
</html>
```

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Save, Help, Outline, Timeline.
- Left Sidebar (EXPLORER):** P29 - JOGO DA VELHA (expanded), # estilos.css, <jogo.html>, p29_jogo_da_velha.png, JS script.js.
- Central Area:** Welcome tab selected, jogo.html tab open.
- Code Content:** An HTML file for a Tic-Tac-Toe game.

```
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
```

61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90

```
</head>
<body>
<!-- A tag &lt;body&gt; contém todo o conteúdo visível de uma página HTML.
    Aqui você define o que os usuários interagem e veem no navegador. --&gt;
<!-- Título--&gt;
&lt;h1&gt;Jogo da Velha&lt;/h1&gt;
<!-- Início do bloco de pontuações --&gt;
&lt;div id="Pontuacoes"&gt;
    <!-- Texto fixo "Jogador: " para indicar a
        pontuação do jogador humano --&gt;
    Jogador:
    <!-- Elemento &lt;span&gt; que contém a pontuação do jogador.
        O 'id' é um identificador único para permitir
        manipulações específicas via CSS ou JavaScript.
        Inicia com 0 e pode ser atualizado dinamicamente. --&gt;
    &lt;span id="Pontuacao-jogador"&gt;0&lt;/span&gt;
    <!-- Separador visual entre as pontuações do
        jogador e do computador --&gt;
    |
    <!-- Texto fixo "Computador: " para indicar a
        pontuação do adversário controlado pelo computador --&gt;
    Computador:</pre>
```
- Bottom Status Bar:** ⌂ 0 △ 0 ⌂ 0, Ln 61, Col 1, Spaces: 8, UTF-8, CRLF, HTML, Go Live, ⌂

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Find, Outline, Timeline.
- Active File:** jogo.html
- Project Explorer:** P29 - JOGO DA VELHA (estilos.css, jogo.html, p29_jogo_da_velha.png, script.js)
- Code Editor:** The code is written in HTML and includes comments in Portuguese. It defines a scoring system and a game board.

```
<!-- Elemento <span> que contém a pontuação do computador.  
Inicia com 0 e pode ser atualizado dinamicamente com scripts. -->  
<span id="pontuacao-computador">0</span>  
  
<!-- Separador visual entre as pontuações do  
computador e os empates -->  
  
<!-- Texto fixo "Empates: " para indicar a  
quantidade de jogos empatados -->  
  
Empates:  
  
<!-- Elemento <span> que contém o número de empates.  
Inicia com 0 e pode ser atualizado dinamicamente com scripts. -->  
<span id="pontuacao-empates">0</span>  
  
</div>  
<!-- Fim do bloco de pontuações -->  
  
<!-- Início do contêiner do tabuleiro do jogo -->  
<div class="tabuleiro" id="tabuleiro">  
  
<!-- Célula do tabuleiro onde ocorre uma jogada ao clicar.  
'onclick' é um evento que chama a função 'fazerJogada(0)'  
quando a célula é clicada.  
  
O argumento '0' refere-se à posição desta
```
- Status Bar:** Ln 91, Col 1, Spaces: 8, UTF-8, CRLF, HTML, Go Live, Notifications.

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Save, Help, Outline, Timeline.
- Left Sidebar (EXPLORER):** P29 - JOGO DA VELHA folder containing # estilos.css, <jogo.html>, p29_jogo_da_velha.png, and JS script.js.
- Active Tab:** jogo.html
- Code Area:** The code is for a Tic-Tac-Toe game tabuleiro (board). It consists of a 3x3 grid of div elements, each with a class of "celula" and an onclick event that calls the "fazerJogada" function with an index parameter (0 through 8).

```
<div class="celula" onclick="fazerJogada(0)"></div>
<div class="celula" onclick="fazerJogada(1)"></div>
<div class="celula" onclick="fazerJogada(2)"></div>
<br>
<div class="celula" onclick="fazerJogada(3)"></div>
<div class="celula" onclick="fazerJogada(4)"></div>
<div class="celula" onclick="fazerJogada(5)"></div>
<br>
```
- Bottom Status Bar:** Line 121, Column 4, Spaces: 8, UTF-8, CRLF, HTML, Go Live, and a bell icon.

The screenshot shows a code editor interface with the following details:

- File Bar:** File Edit Selection View Go Run ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Save, Find, Outline, Timeline.
- Left Sidebar:** Shows a tree view under "P29 - JOGO DA VELHA" with files: # estilos.css, <> jogo.html (selected), p29_jogo_da_velha.png, and JS script.js.
- Central Area:** The "jogo.html" tab is active, displaying the following HTML code:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Jogo da Velha</title>
    <link href="estilos.css" rel="stylesheet">
  </head>
  <body>
    <div id="tabuleiro">
      <!-- Sétima célula do tabuleiro, com evento de clique que chama 'fazerJogada(6)'. -->
      <div class="celula" onclick="fazerJogada(6)"></div>

      <!-- Oitava célula do tabuleiro, com evento de clique que chama 'fazerJogada(7)'. -->
      <div class="celula" onclick="fazerJogada(7)"></div>

      <!-- Nona e última célula do tabuleiro, com evento de clique que chama 'fazerJogada(8)'. -->
      <div class="celula" onclick="fazerJogada(8)"></div>

    </div>
    <!-- Fim do contêiner do tabuleiro -->

    <!-- Inclusão de um arquivo JavaScript externo chamado 'script.js'. Este script é responsável pela lógica do jogo, manipulação dos eventos de clique, e atualização da interface do jogo. -->
    <script src="script.js"></script>
  </body>
</html>
```
- Bottom Status Bar:** Shows file statistics: 0 0 0, Ln 151, Col 9, Spaces: 8, UTF-8, CRLF, HTML, Go Live, and a refresh icon.

A screenshot of a code editor interface, likely Microsoft Visual Studio Code, displaying a CSS file named `# estilos.css`. The file contains styles for a Tic-Tac-Toe game, including styling for the body and a tabuleiro class.

```
/* Estilizações para o corpo do documento HTML */
body {
    font-family: Arial, sans-serif;
    /* Define a família da fonte como Arial, com um
       fallback para qualquer fonte sans-serif disponível */
    text-align: center;
    /* Alinha todo o texto contido no corpo do
       documento ao centro */
    margin-top: 50px;
    /* Aplica uma margem superior de 50 pixels para separar o
       conteúdo do topo da página */
}

/* Estilizações para a classe .tabuleiro, usada para
   posicionar o tabuleiro de jogo */
.tabuleiro {
    margin: 0 auto;
    /* Define as margens laterais como automáticas para
       centralizar o tabuleiro horizontalmente na página,
       e a margem superior e inferior como 0 */
    display: inline-block;
    /* Define o tipo de exibição como inline-block,
       permitindo que o tabuleiro seja alinhado como um elemento inline
       mas respeitando as propriedades de box-model como um elemento block */
```

A screenshot of a code editor interface, likely Visual Studio Code, displaying a CSS file for a Tic-Tac-Toe game. The left sidebar shows files: 'jogo.html', '# estilos.css', 'p29_jogo_da_velha.png', and 'script.js'. The top bar includes standard menu items like File, Edit, Selection, View, Go, Run, and a search bar. The main area shows the following CSS code:

```
31 }  
32 }  
33 /* Estilizações para a classe .celula, usada para cada  
34 |   célula individual no tabuleiro de jogo */  
35 .celula {  
36     width: 100px;  
37     /* Define a largura da célula como 100 pixels */  
38     height: 100px;  
39     /* Define a altura da célula como 100 pixels */  
40     font-size: 48px;  
41     /* Define o tamanho da fonte dentro da célula como 48 pixels,  
42     |   útil para exibir grandes caracteres (X ou O) */  
43     cursor: pointer;  
44     /* Muda o cursor para um ponteiro quando o mouse passa  
45     |   sobre a célula, indicando que é clicável */  
46     border: 2px solid #ccc;  
47     /* Adiciona uma borda sólida com 2 pixels de espessura e  
48     |   cor cinza claro (#ccc) ao redor da célula */  
49     display: inline-flex;  
50     /* Usa o modelo de layout flexível em linha, permitindo que o  
51     |   conteúdo da célula seja flexível e alinhado facilmente */  
52     justify-content: center;
```

The screenshot shows a code editor interface with the following details:

- File Bar:** File Edit Selection View Go Run ...
- Search Bar:** P29 - Jogo da Velha
- Sidebar:** EXPLORER, P29 - JOGO DA VELHA (containing # estilos.css, jogo.html, p29_jogo_da_velha.png, script.js), OUTLINE, and TIMELINE.
- Code Area:** The file # estilos.css is open, showing the following CSS code for a Tic-Tac-Toe board:

```
/* Alinha o conteúdo da célula horizontalmente ao centro,
   parte do modelo flexível */
align-items: center;
/* Alinha o conteúdo da célula verticalmente ao centro,
   também parte do modelo flexível */
vertical-align: middle;
/* Garante que o alinhamento vertical do elemento
   inline-flex (a célula) seja ao meio em relação
   ao seu contêiner */
margin: 5px;
/* Adiciona uma margem de 5 pixels ao redor de cada célula
   para garantir algum espaço entre as células adjacentes */
}

/* Estilizações para o pseudo-elemento :hover da classe .celula,
   aplicadas quando o cursor do mouse passa sobre a célula */
.celula:hover {
    background-color: #1a1da7;
    /* Muda a cor de fundo da célula para um azul muito
       escuro (#1a1da7) quando o mouse passa sobre ela,
       indicando que a célula é interativa e realçando a
       célula sob o cursor */
    color: #ccc;
}
```

The code uses CSS Flexbox properties to center the content both horizontally and vertically within each cell. It also includes a hover state for the cells, changing their background color to a dark blue (#1a1da7) and keeping the text color white (#ccc).

The screenshot shows a dark-themed code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Problems, Editor, Outline, Timeline.
- Left Sidebar (EXPLORER):** P29 - JOGO DA VELHA folder containing # estilos.css, jogo.html, p29_jogo_da_velha.png, and script.js.
- Active File:** # estilos.css (Line 91)
- Code Content:**

```
91 }  
92 }  
93  
94 /* Estilizações para o elemento com ID #Pontuacoes, usado  
95 para exibir as pontuações no jogo */  
96 #Pontuacoes {  
97  
98     margin-top: 20px;  
99     /* Adiciona uma margem superior de 20 pixels, criando  
100    |    espaço entre as pontuações e outros elementos  
101    |    acima delas */  
102  
103    font-weight: bold;  
104    /* Define o peso da fonte para negrito, destacando  
105    |    visualmente o texto das pontuações para  
106    |    fácil leitura */  
107  
108 }
```

- Bottom Status Bar:** ⌂ 0 △ 0 ⌂ 0, Ln 91, Col 1, Spaces: 8, UTF-8, CRLF, CSS, Go Live, ⌁

A screenshot of a code editor interface, likely Microsoft Visual Studio Code, displaying a JavaScript file named `script.js`. The file contains the code for a Tic-Tac-Toe game logic.

The code defines variables for the board, current player, active game status, and scores for both player and computer. It also initializes a variable for ties.

```
// Cria uma variável chamada 'tabuleiro' e a inicializa
// com um array de 9 elementos vazios.
// Cada elemento do array representa uma célula no
// tabuleiro do jogo da velha.
let tabuleiro = ['', '', '', '', '', '', '', '', ''];

// Define a variável 'jogadorAtual' e atribui o valor 'X',
// indicando que o jogador 'X' será o primeiro a jogar.
let jogadorAtual = 'X';

// Inicializa a variável 'jogoAtivo' como true, indicando
// que o jogo está ativo e pode prosseguir.
let jogoAtivo = true;

// Define e inicializa a variável 'pontuacaoJogador' com 0.
// Esta variável armazena a pontuação do jogador humano
// durante a sessão de jogo.
let pontuacaoJogador = 0;

// Define e inicializa a variável 'pontuacaoComputador' com 0.
// Esta variável armazena a pontuação do computador
// durante a sessão de jogo.
let pontuacaoComputador = 0;

// Define e inicializa a variável 'pontuacaoEmpates' com 0.
// Esta variável conta o número de jogos que
// terminaram em empate.
let pontuacaoEmpates = 0;
```

P29 - Jogo da Velha

```
// Cria uma constante chamada 'COMBINACOES_VITORIA' e a
// atribui a um array de arrays.
// Cada sub-array contém os índices das posições no
// tabuleiro que formam uma linha, coluna ou diagonal completa.
const COMBINACOES_VITORIA = [
    [0, 1, 2], // Primeira linha: estes índices representam as células na primeira linha do tabuleiro de jogo da
               // velha.
    [3, 4, 5], // Segunda linha: estes índices representam as células na segunda linha do tabuleiro.
    [6, 7, 8], // Terceira linha: estes índices representam as células na terceira linha do tabuleiro.

    [0, 3, 6], // Primeira coluna: estes índices representam as células na primeira coluna vertical do tabuleiro.
    [1, 4, 7], // Segunda coluna: estes índices representam as células na segunda coluna vertical.
    [2, 5, 8], // Terceira coluna: estes índices representam as células na terceira coluna vertical.

    [0, 4, 8], // Diagonal principal: estes índices vão do canto superior esquerdo ao canto inferior direito do
               // tabuleiro.
    [2, 4, 6] // Diagonal secundária: estes índices vão do canto superior direito ao canto inferior esquerdo.

];
// Define a função 'fazerJogada', que é responsável por
// processar uma jogada feita em uma célula específica do tabuleiro.
function fazerJogada(indiceCelula) {
    // Esta linha de código verifica duas condições antes
    // de proceder com a jogada:
    // 1. Se 'jogoAtivo' é falso, o que indica que o jogo não
               // está mais em progresso (talvez alguém já tenha
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) window with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Find, Outline, Timeline.
- Active File:** script.js (highlighted in yellow)
- Content:** A snippet of JavaScript code for a Tic-Tac-Toe game. The code includes comments in Portuguese explaining the logic of the 'fazerJogada' function, such as checking for wins and updating the board state.
- Bottom Status Bar:** Line 59, Column 4, Spaces: 4, UTF-8, CRLF, {} JavaScript, Go Live, Timeline.

```
// ganho ou o jogo acabou em empate),
// a função irá parar imediatamente e não executará mais nada.
// 2. Se a célula no 'tabuleiro' no índice especificado
// 'indiceCelula' já contém um valor (ou seja,
// não está vazia, indicado por ''),
// a função também irá parar. Isso evita que uma célula
// já ocupada seja sobreescrita, o que poderia
// corromper o estado do jogo.
if (!jogoAtivo || tabuleiro[indiceCelula] !== '') return;
// A palavra-chave 'return' é usada aqui para sair da função
// sem fazer mais nada se qualquer uma das condições acima for verdadeira.

// Atribui o símbolo do jogador atual ('X' ou 'O') à
// célula do tabuleiro no índice especificado.
// Isso efetivamente faz a jogada colocando o símbolo
// do jogador na posição escolhida.
tabuleiro[indiceCelula] = jogadorAtual;

// Chama a função 'renderizarTabuleiro' para atualizar
// visualmente o tabuleiro na página,
// refletindo a jogada que acabou de ser feita.
renderizarTabuleiro();

// Verifica se a jogada resultou em uma vitória.
// A função 'verificarVitoria' checa todas as possíveis
// combinações de vitória para ver se o jogador atual formou uma linha.
if (verificarVitoria()) {

    // Se uma vitória foi detectada, define 'jogoAtivo' como false,
```

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Find, Outline, Timeline.
- Active File:** script.js
- Content:** A JavaScript function named `fazerJogada` is shown. The code handles the final logic of a game round, including marking the board as finished, updating player scores, pausing briefly, displaying a victory alert, and restarting the game. It ends with a `return` statement.

```
// indicando que o jogo terminou e não devem ser
// permitidas mais jogadas.
jogoAtivo = false;

// Atualiza as pontuações dos jogadores, incrementando a
// pontuação do jogador atual.
atualizarPontuacoes(jogadorAtual);

// Usa 'setTimeout' para criar um pequeno atraso antes
// de mostrar o alerta e reiniciar o jogo.
// Isso é feito para garantir que o estado final do
// jogo seja visível antes de qualquer interação.
setTimeout(() => {

    // Mostra um alerta informando que o jogador atual venceu.
    alert(`#${jogadorAtual} venceu!`);

    // Chama a função 'reiniciarJogo' para resetar o
    // tabuleiro e todas as variáveis relevantes,
    // começando um novo jogo.
    reiniciarJogo();

}, 100); // O atraso é de 100 milissegundos.

// A palavra-chave 'return' é usada para sair da
// função imediatamente após uma vitória,
// garantindo que nenhuma outra lógica ou jogada
// seja processada após o término do jogo.
return;
```

- Status Bar:** Line 89, Col 7, Spaces: 4, UTF-8, CRLF, {} JavaScript, Go Live, Timeline.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The title bar reads "P29 - Jogo da Velha". The left sidebar contains icons for Explorer, Search, Problems, and Outline/Timeline. The Explorer view shows a project folder "P29 - JOGO DA VELHA" containing files: "# estilos.css", "jogo.html", "p29_jogo_da_velha.png", and "script.js". The "script.js" file is currently open in the main editor area. The code is written in JavaScript and handles the logic for a Tic-Tac-Toe game, specifically checking for a draw (empate). The code includes comments in Portuguese explaining the logic.

```
119     }
120
121
122     // Inicia uma condicional para verificar se o
123     // jogo terminou em empate.
124     // A função 'verificarEmpate' checa se todas as células do
125     // tabuleiro estão preenchidas e se não há uma vitória.
126     if (verificarEmpate()) {
127
128         // Se um empate é detectado, a variável 'jogoAtivo' é definida como false.
129         // Isso indica que o jogo terminou e novas jogadas não são permitidas.
130         jogoAtivo = false;
131
132         // Chama a função 'atualizarPontuacoes' com o argumento 'empate'.
133         // Esta função incrementa a pontuação de empates e
134         // atualiza a visualização das pontuações.
135         atualizarPontuacoes('empate');
136
137         // Utiliza 'setTimeout' para criar um pequeno atraso
138         // antes de mostrar um alerta e reiniciar o jogo.
139         // Esse atraso permite que os jogadores vejam o estado
140         // final do tabuleiro antes do alerta ser mostrado.
141         setTimeout(() => {
142
143             // Mostra um alerta na tela com a mensagem 'Empate!',
144             // indicando o resultado do jogo.
145             alert('Empate!');
146
147             // Chama a função 'reiniciarJogo' que prepara o
148             // tabuleiro e as variáveis para um novo jogo,
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top bar includes standard menu items: File, Edit, Selection, View, Go, Run, and a Help icon. To the right of the menu is a search bar containing the text "P29 - Jogo da Velha". On the far right are icons for closing, minimizing, maximizing, and exiting the window.

The left sidebar features several icons: a clipboard (EXPLORER), a magnifying glass (SEARCH), a gear (SETTINGS), a person (USERS), and a timeline (TIMELINE). Below these are sections for "OUTLINE" and "TIMELINE".

The main workspace displays a code editor with four tabs: "jogo.html", "# estilos.css", "script.js" (which is currently active), and "x". The "script.js" tab contains the following code:

```
// resetando tudo para o estado inicial.  
reiniciarJogo();  
  
, 100); // O atraso é de 100 milissegundos.  
  
// A palavra-chave 'return' é usada para sair da  
// função imediatamente após detectar um empate,  
// garantindo que nenhuma outra lógica ou jogada seja processada.  
return;  
  
}  
  
// Alterna o jogador atual. Se o jogador atual é 'X',  
// muda para 'O', e vice-versa.  
// Isso é feito usando um operador ternário: se a  
// condição 'jogadorAtual === 'X'' é verdadeira,  
// o jogadorAtual será definido como 'O'. Se for falsa,  
// será definido como 'X'.  
jogadorAtual = jogadorAtual === 'X' ? 'O' : 'X';  
  
// Inicia uma condição para verificar se o jogador  
// atual é 'O' e se o jogo ainda está ativo.  
// Essa condição geralmente é usada para permitir que o  
// computador faça sua jogada automaticamente  
// quando é sua vez, simulando um jogo contra a máquina.  
if (jogadorAtual === 'O' && jogoAtivo) {  
  
    // Utiliza a função 'setTimeout' para criar um  
    // pequeno atraso antes da jogada do computador.
```

The status bar at the bottom shows the current file is "script.js", line 149, column 6, with 4 spaces, using UTF-8 encoding, and is a CRLF file. It also indicates the code is in "JavaScript" mode and has "Go Live" options.

The screenshot shows a Microsoft Visual Studio Code (VS Code) window with the title bar "P29 - Jogo da Velha". The left sidebar contains icons for Explorer, Search, Open, and Outline/Timeline. The Explorer section shows a folder named "P29 - JOGO DA VELHA" containing files: "# estilos.css", "jogo.html", "p29_jogo_da_velha.png", and "script.js". The "script.js" file is currently open in the main editor area.

The code in "script.js" is written in JavaScript and handles the logic for a computer player in a Tic-Tac-Toe game. It includes comments explaining the purpose of each section:

```
// Isso pode ser útil para simular tempo de
// pensamento e tornar o jogo mais realista.
setTimeout(() => {
    // Chama a função 'movimentoComputador', que
    // contém a lógica de como o computador escolhe sua jogada.
    // Esta função vai determinar a melhor jogada possível
    // baseada na lógica implementada (por exemplo, um algoritmo minimax).
    movimentoComputador();
}, 500); // O atraso é de 500 milissegundos (meio segundo).

}

// Define a função 'movimentoComputador', que controla
// como o computador decide sua jogada.
function movimentoComputador() {
    // Inicializa a variável 'melhorPontuacao' com -Infinity,
    // que será usada para comparar e encontrar a melhor
    // pontuação possível entre as jogadas.
    let melhorPontuacao = -Infinity;

    // Declara a variável 'movimento', que armazenará o
    // índice da melhor jogada encontrada para o computador.
    let movimento;

    // Inicia um loop para percorrer cada célula do tabuleiro.
```

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Left Sidebar (EXPLORER):** P29 - JOGO DA VELHA (expanded), showing files: # estilos.css, jogo.html, p29_jogo_da_velha.png, script.js.
- Active Tab:** script.js
- Code Area:** The script.js file contains the following code:

```
for (let i = 0; i < tabuleiro.length; i++) {
    // Verifica se a célula atual está vazia, indicando
    // que é um lugar válido para fazer uma jogada.
    if (tabuleiro[i] === '') {
        // Temporariamente marca a célula com 'O', que é
        // o símbolo usado pelo computador.
        tabuleiro[i] = 'O';

        // Chama a função 'minimax' para avaliar a
        // pontuação dessa configuração de tabuleiro,
        // passando o tabuleiro atual, uma profundidade de
        // recursão inicial (0) e o indicador de que é a
        // vez do minimizador (false).
        let pontuacao = minimax(tabuleiro, 0, false);

        // Restaura a célula para vazia após a avaliação,
        // pois ainda não é a jogada final.
        tabuleiro[i] = '';

        // Compara a pontuação retornada com a melhor
        // pontuação já encontrada.
        if (pontuacao > melhorPontuacao) {
            // Se a nova pontuação for maior, atualiza
            // 'melhorPontuacao' com este novo valor.
            melhorPontuacao = pontuacao;
            // Registra o índice desta célula como a
        }
    }
}
```

Bottom Status Bar: Ln 209, Col 1, Spaces: 4, UTF-8, CRLF, {}, JavaScript, Go Live

The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Toolbar:** Minimize, Maximize, Close
- Left Sidebar (EXPLORER):** P29 - JOGO DA VELHA folder containing # estilos.css, jogo.html, p29_jogo_da_velha.png, and script.js.
- Current File:** script.js (selected tab)
- Code Content:** A snippet of JavaScript code for a minimax algorithm in a Tic-Tac-Toe game. The code includes comments explaining the logic of returning scores based on the winning player ('O' or 'X') and the depth of the search. It also shows the implementation of the minimax algorithm, including initializing the best score to negative infinity, iterating over the board cells, and verifying if a cell is empty.
- Bottom Status Bar:** Line 269, Column 1, Spaces: 4, UTF-8, CRLF, {} JavaScript, Go Live, Timeline

```
// Retorna uma pontuação baseada no jogador que venceu.  
// Se 'O' é o vencedor, retorna 10 menos a profundidade.  
// Isso significa que quanto mais rápido o computador vencer,  
// melhor é a pontuação, pois o objetivo é maximizar a pontuação  
// do 'O' e minimizar a quantidade de movimentos até a vitória.  
// Se 'X' (o humano) venceu, retorna a profundidade menos 10,  
// indicando uma situação desfavorável para o computador.  
// A profundidade subtraída implica que uma derrota mais tardia é  
// menos prejudicial do que uma derrota imediata.  
return resultado === 'O' ? 10 - profundidade : profundidade - 10;  
  
}  
  
// Se 'estaMaximizando' for verdadeiro, estamos  
// avaliando os movimentos para o computador 'O'.  
if (estaMaximizando) {  
  
    // Inicializa 'melhorPontuacao' como -Infinity para  
    // garantir que qualquer pontuação inicial será maior.  
    let melhorPontuacao = -Infinity;  
  
    // Itera sobre cada célula do tabuleiro.  
    for (let i = 0; i < tabuleiro.length; i++) {  
  
        // Verifica se a célula atual do tabuleiro está vazia, o  
        // que significa que ninguém jogou nessa posição.  
        // Esta é uma verificação crucial porque só se pode jogar em  
        // células que ainda não foram ocupadas.
```

The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Toolbar:** Standard window control buttons (Minimize, Maximize, Close) and a Go Live button.
- Left Sidebar (EXPLORER):** Shows the project structure under "P29 - JOGO DA VELHA": # estilos.css, jogo.html, p29_jogo_da_velha.png, and script.js (selected).
- Central Area:** A tab bar with tabs: jogo.html, # estilos.css, JS script.js (highlighted), and X. Below the tabs, the code for the script.js file is displayed.
- Code Content:** The code implements a minimax algorithm for a Tic-Tac-Toe board represented by an array "tabuleiro". The code includes comments explaining the logic of simulating moves, calculating scores, and undoing moves to maintain consistency across simulations.
- Bottom Status Bar:** Includes icons for file operations, a search icon, line number (Ln 299, Col 1), character count (Spaces: 4), encoding (UTF-8), line endings (CRLF), file type ({} JavaScript), and Go Live.

```
if (tabuleiro[i] === '') {  
    // Simula uma jogada colocando o símbolo 'O' na célula atual.  
    // Esta linha altera temporariamente o tabuleiro,  
    // adicionando o 'O' na posição i.  
    // Isso é feito para avaliar o impacto potencial  
    // dessa jogada nas próximas etapas do jogo.  
    tabuleiro[i] = 'O';  
  
    // Chama a função 'minimax' recursivamente para avaliar o  
    // tabuleiro depois da jogada simulada.  
    // A profundidade é incrementada por 1, indicando que  
    // estamos um nível mais profundo na árvore de jogadas.  
    // O parâmetro 'false' significa que agora é a vez do  
    // minimizador (o jogador humano, que usa 'X'),  
    // para jogar, refletindo a alternância de turnos no jogo.  
    let pontuacao = minimax(tabuleiro, profundidade + 1, false);  
  
    // Desfaz a jogada simulada, removendo o 'O' e  
    // retornando a célula ao estado vazio.  
    // Isso é necessário porque a mesma célula será  
    // considerada em futuras simulações de jogadas,  
    // e para cada simulação, o tabuleiro precisa estar no  
    // estado que estava antes da jogada ser feita.  
    tabuleiro[i] = '';  
  
    // Compara a pontuação obtida da simulação (que representa o  
    // resultado de um conjunto de jogadas até o final do jogo  
    // a partir dessa posição) com a melhor pontuação que já  
    // foi encontrada para outras simulações no mesmo nível de profundidade.
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) window with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Find, Outline, Timeline.
- File Explorer (Left):** Shows a folder named "P29 - JOGO DA VELHA" containing files: # estilos.css, jogo.html, p29_jogo_da_velha.png, and script.js (which is currently selected).
- Text Editor (Main Area):** Displays a JavaScript file named "script.js". The code implements a minimax algorithm for a Tic-Tac-Toe game. It includes comments explaining the logic of maximizing and minimizing scores, initializing variables, iterating over board cells, and checking for empty cells.
- Bottom Status Bar:** Shows file statistics: 0 0 0, and navigation information: Ln 329, Col 7, Spaces: 4, UTF-8, CRLF, {}, JavaScript, Go Live.

```
// 'Math.max' é usado para manter a melhor pontuação possível
// para o maximizador, que neste caso é o computador jogando com 'O'.
// Isso significa que o algoritmo está buscando a jogada que
// leva ao melhor resultado possível assumindo que o oponente
// também joga da melhor forma possível.
melhorPontuacao = Math.max(pontuacao, melhorPontuacao);

}

}

// Retorna a melhor pontuação encontrada para
// essa ramificação de jogadas.
return melhorPontuacao;

} else {

    // Inicializa 'melhorPontuacao' como Infinity
    // para maximizar a pontuação do oponente.
    // Ao começar com Infinity, qualquer pontuação mais
    // baixa encontrada durante a simulação das jogadas será considerada,
    // pois buscamos a menor pontuação, que representa a
    // melhor jogada para o oponente e a pior para o computador.
    let melhorPontuacao = Infinity;

    // Itera sobre cada célula do tabuleiro. Esta iteração
    // permite explorar todas as possíveis jogadas que o oponente pode fazer.
    for (let i = 0; i < tabuleiro.length; i++) {
        // Verifica se a célula atual está vazia. Uma célula
```

A screenshot of a code editor interface, likely Visual Studio Code, displaying a JavaScript file named `script.js` for a game titled "P29 - Jogo da Velha".

The code implements the Minimax algorithm to find the best move for the human player ('X'). It includes comments explaining the logic, such as simulating a move by the opponent ('O') and updating the board state.

```
// vazia indica que é um local disponível para fazer uma jogada.
if (tabuleiro[i] === '') {
    // Faz uma jogada simulada colocando 'X' na
    // célula, que é o símbolo do oponente humano.
    // Esta simulação é crucial para avaliar o impacto
    // potencial dessa jogada no resultado do jogo.
    tabuleiro[i] = 'X';

    // Chama 'minimax' recursivamente, mas agora com o
    // objetivo de maximizar a pontuação na próxima chamada,
    // pois a função alternará de volta para a perspectiva
    // do computador, que tentará maximizar sua pontuação.
    // A profundidade é incrementada por 1, significando
    // que estamos um nível mais profundo na árvore de decisão.
    let pontuacao = minimax(tabuleiro, profundidade + 1, true);

    // Desfaz a jogada simulada revertendo a célula para o estado vazio.
    // Este passo é essencial para manter o estado original
    // do tabuleiro, permitindo que futuras simulações sejam precisas.
    tabuleiro[i] = '';

    // Atualiza 'melhorPontuacao' para o menor valor
    // entre a pontuação obtida e a melhor pontuação anterior.
    // Usando 'Math.min', estamos buscando a menor
    // pontuação possível, o que equivale à melhor jogada para o humano e,
    // consequentemente, à pior para o computador. Este
    // processo é o cerne do minimizar no algoritmo minimax,
    // onde o objetivo é reduzir o sucesso potencial do adversário.
    melhorPontuacao = Math.min(pontuacao, melhorPontuacao);
}
```


The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Find, Outline, Timeline.
- File Explorer (Left):** P29 - JOGO DA VELHA folder containing # estilos.css, jogo.html, p29_jogo_da_velha.png, and script.js.
- Active Tab:** script.js
- Code Area:** The code is written in JavaScript and defines a function to check for a winner in a 3x3 grid.

```
// Esta função verifica se alguém venceu o jogo.
// Ela recebe uma matriz 3x3 de símbolos ('X' ou 'O') e retorna o símbolo do vencedor ou null se não houver vencedor.
function verificarVencedor(tabuleiro) {
    // Verifica linhas
    for (let i = 0; i < 3; i++) {
        if (tabuleiro[i][0] === tabuleiro[i][1] && tabuleiro[i][1] === tabuleiro[i][2]) {
            return tabuleiro[i][0];
        }
    }

    // Verifica colunas
    for (let j = 0; j < 3; j++) {
        if (tabuleiro[0][j] === tabuleiro[1][j] && tabuleiro[1][j] === tabuleiro[2][j]) {
            return tabuleiro[0][j];
        }
    }

    // Verifica diagonais
    if (tabuleiro[0][0] === tabuleiro[1][1] && tabuleiro[1][1] === tabuleiro[2][2]) {
        return tabuleiro[0][0];
    }

    if (tabuleiro[0][2] === tabuleiro[1][1] && tabuleiro[1][1] === tabuleiro[2][0]) {
        return tabuleiro[0][2];
    }

    // Se o loop termina sem encontrar uma combinação vencedora, retorna null.
    return null;
}

// Define a função 'verificarVitoria', que é usada
// para determinar se há uma vitória no jogo.
function verificarVitoria() {
    const tabuleiro = [
        [null, null, null],
        [null, null, null],
        [null, null, null]
    ];

    const vencedor = verificarVencedor(tabuleiro);
    if (vencedor) {
        console.log(`O vencedor é ${vencedor}`);
    } else {
        console.log('Ninguém venceu');
    }
}
```

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Find, Outline, Timeline.
- File Explorer (Left):** P29 - JOGO DA VELHA folder containing # estilos.css, jogo.html, p29_jogo_da_velha.png, and script.js.
- Active Tab:** script.js
- Code Content:** The script.js file contains the following code:

```
// A função 'verificarVitoria' retorna true se
// 'verificarVencedor' retornar algo diferente de null,
// o que significa que um jogador venceu. Caso contrário, retorna false.
return verificarVencedor() !== null;

}

// Define a função 'verificarEmpate', utilizada
// para determinar se o jogo resultou em empate.
function verificarEmpate() {

    // Utiliza o método 'includes' do array para
    // verificar se o tabuleiro ainda contém alguma célula vazia ('').
    // Se todas as células estiverem preenchidas (não incluir ''),
    // isso indica que o tabuleiro está completo
    // e não há espaços vazios restantes para jogadas adicionais.
    // A função retorna true se não houver células
    // vazias (indicando um possível empate),
    // e false se ainda houver espaços onde os jogadores podem jogar.
    return !tabuleiro.includes('');
}

// Define a função 'atualizarPontuacoes', que é responsável
// por atualizar as pontuações do jogo com base no resultado de cada rodada.
function atualizarPontuacoes(vencedor) {

    // Verifica se o resultado da rodada foi um 'empate'.
    if (vencedor === 'empate') {
```
- Bottom Status Bar:** ⌘ 0 ⌛ 0 ⌂ 0, Line 449, Col 4, Spaces: 4, UTF-8, CRLF, {}, JavaScript, Go Live, ⌓

The screenshot shows a Microsoft Visual Studio Code (VS Code) window with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Find, Outline, Timeline.
- Active File:** script.js (highlighted in blue)
- Content:** A snippet of JavaScript code for a Tic-Tac-Toe game, specifically handling score updates based on the game result ('X' or 'O').

```
// Se for um empate, incrementa o contador de empates.  
// 'pontuacaoEmpates' é uma variável que rastreia o  
// número total de jogos que terminaram sem um vencedor claro.  
pontuacaoEmpates++;  
  
} else if (vencedor === 'X') {  
  
    // Caso o vencedor seja o jogador 'X', incrementa  
    // a pontuação do jogador.  
    // 'pontuacaoJogador' é uma variável que mantém a  
    // contagem das vitórias do jogador humano.  
    pontuacaoJogador++;  
  
} else {  
  
    // Se o vencedor for 'O', que representa o computador  
    // neste contexto, incrementa a pontuação do computador.  
    // 'pontuacaoComputador' rastreia quantas vezes o computador ganhou.  
    pontuacaoComputador++;  
  
}  
  
// Após atualizar a pontuação adequada, chama a  
// função 'renderizarPontuacoes'.  
// Esta função é responsável por atualizar a interface do  
// usuário para refletir as novas pontuações,  
// garantindo que os jogadores possam ver o estado  
// atual das pontuações em tempo real.  
renderizarPontuacoes();
```

Bottom status bar: ⌂ 0 △ 0 ⌂ 0 Ln 479, Col 1 Spaces: 4 UTF-8 CRLF {} JavaScript ⌂ Go Live ⌂

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Find, Outline, Timeline.
- Active File:** script.js
- Content:** A JavaScript function named renderizarPontuacoes() that updates the user interface with player scores and draw counts.

```
EXPLORER      ...
P29 - JOGO DA VELHA
# estilos.css
jogo.html
p29_jogo_da_velha.png
JS script.js

jogo.html   # estilos.css  JS script.js  X
JS script.js > atualizarPontuacoes

509 }
510
511
512 // Define a função 'renderizarPontuacoes', responsável por
513 // atualizar a visualização das pontuações na interface do usuário.
514 function renderizarPontuacoes() {
515
516     // Acessa o elemento HTML com o id 'pontuacao-jogador'.
517     // 'document.getElementById' é um método do DOM (Document
518     // Object Model) que retorna o elemento que possui o ID especificado.
519     // Uma vez que o elemento é retornado, a propriedade
520     // 'textContent' é usada para modificar o texto contido nesse elemento.
521     // A pontuação atual do jogador humano é atribuída a esse
522     // elemento, assim atualizando o valor visível na tela.
523     document.getElementById('pontuacao-jogador').textContent = pontuacaoJogador;
524
525     // Similar ao passo anterior, acessa o elemento com o id 'pontuacao-computador'.
526     // Atualiza o texto desse elemento para refletir a pontuação atual do computador,
527     // permitindo que os usuários vejam quantas vezes o computador ganhou.
528     document.getElementById('pontuacao-computador').textContent = pontuacaoComputador;
529
530     // Acessa o elemento com o id 'pontuacao-empates'.
531     // Atualiza o conteúdo textual deste elemento para mostrar o
532     // número de empates ocorridos até agora no jogo.
533     // Isso fornece um feedback completo sobre todos os
534     // resultados possíveis das partidas jogadas.
535     document.getElementById('pontuacao-empates').textContent = pontuacaoEmpates;
536
537 }
538
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) window with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Find, Outline, Timeline.
- Active File:** script.js (highlighted in blue)
- Content:** A snippet of JavaScript code for rendering a Tic-Tac-Toe board based on a 2D array.

```
// Define a função 'renderizarTabuleiro', que atualiza o
// tabuleiro visual na interface do usuário
// com base no estado atual do array 'tabuleiro', que
// contém os símbolos das jogadas ('X', 'O', ou vazio).
function renderizarTabuleiro() {
    // Inicia um loop que percorrerá cada elemento do array 'tabuleiro'.
    // 'tabuleiro.length' dá o número total de células no
    // tabuleiro, que é 9 num jogo da velha padrão (3x3).
    for (let i = 0; i < tabuleiro.length; i++) {
        // Acessa cada célula visual correspondente no HTML pelo
        // índice usando 'document.getElementsByClassName('celula')[i]'.
        // 'getElementsByClassName' retorna uma coleção de todos
        // elementos que possuem a classe especificada, neste caso, 'celula'.
        // '[i]' acessa o elemento da coleção correspondente ao
        // índice atual do loop, sincronizando os índices do array 'tabuleiro'
        // com os elementos da interface do usuário.
        const celula = document.getElementsByClassName('celula')[i];
        // Define o conteúdo de texto de cada célula visual no tabuleiro
        // para corresponder ao valor da célula no array 'tabuleiro'.
        // Se 'tabuleiro[i]' é 'X', 'O' ou '', o mesmo será mostrado
        // na célula correspondente na interface do usuário.
        // 'textContent' é uma propriedade que define ou retorna o
        // conteúdo de texto de um nó e de seus descendentes.
        celula.textContent = tabuleiro[i];
    }
}
```

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Icons:** Explorer, Search, Open, Find, Outline, Timeline.
- Active File:** script.js
- Content:** A JavaScript function named 'reiniciarJogo' is shown. The function initializes a 9-element array 'tabuleiro' with empty strings, sets 'jogadorAtual' to 'X', and sets 'jogoAtivo' to true. It then calls the 'renderizarTabuleiro' function to update the visual representation of the board.

```
569 }  
570  
571  
572 // Define a função 'reiniciarJogo', que é responsável por  
573 // resetar todas as variáveis e estados para seus valores iniciais,  
574 // permitindo que um novo jogo seja iniciado sem  
575 // interferência do estado anterior.  
576 function reiniciarJogo() {  
577  
578     // Redefine o array 'tabuleiro' para um novo  
579     // array de 9 strings vazias.  
580     // Isso limpa o tabuleiro, removendo todas as marcas  
581     // de 'X' e 'O' que foram feitas na partida anterior.  
582     tabuleiro = ['', '', '', '', '', '', '', '', ''];  
583  
584     // Define 'jogadorAtual' para 'X', garantindo que o  
585     // jogador 'X' sempre comece o jogo.  
586     // Esta é uma convenção comum em jogos da velha, onde um  
587     // dos jogadores (neste caso, 'X') sempre inicia a partida.  
588     jogadorAtual = 'X';  
589  
590     // Define a variável 'jogoAtivo' como true, indicando que o  
591     // jogo está ativo e que as jogadas podem ser feitas.  
592     // Isso é importante para permitir que o controle de fluxo do  
593     // jogo continue, como verificar vitórias ou empates.  
594     jogoAtivo = true;  
595  
596     // Chama a função 'renderizarTabuleiro' para  
597     // atualizar visualmente o tabuleiro.  
598     // Isso assegura que o tabuleiro visual reflita o novo
```

- Status Bar:** Line 569, Col 1, Spaces: 4, UTF-8, CRLF, {} JavaScript, Go Live, Timeline.

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P29 - Jogo da Velha
- Toolbar:** Minimize, Maximize, Close, Fullscreen, ...
- Left Sidebar (EXPLORER):**
 - P29 - JOGO DA VELHA
 - # estilos.css
 - jogo.html
 - p29_jogo_da_velha.png
 - script.js
- Central Area:** Script.js tab selected.

```
// estado limpo do array 'tabuleiro',
// mostrando um tabuleiro vazio pronto para uma nova partida.
reiniciarJogo();

// Chama a função 'renderizarPontuacoes' imediatamente.
// Isso é útil para garantir que as pontuações mostradas
// na tela sejam atualizadas para refletir quaisquer mudanças
// que possam ter acontecido no final do jogo anterior,
// como atualizar a contagem de empates ou pontuações de jogadores.
renderizarPontuacoes();
```
- Bottom Bar:** Icons for Outline, Timeline, and Go Live.
- Bottom Status:** Line 599, Col 6, Spaces: 4, UTF-8, CRLF, {} JavaScript, Go Live, and a bell icon.

UC: Projetos

DIAS: 25/10/2024

Assunto:

Small Projects

P30 – Jogo da Forca

The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Toolbar:** Minimize, Maximize, Close, Fullscreen, ...
- Left Sidebar (EXPLORER):**
 - P30 - JOGO DA FORCA
 - index.html
 - p29_jogo_da_forca.png
 - script.js
 - styles.css
- Central Editor Area:** Shows the content of index.html with line numbers and comments explaining the code.

```
1  <!DOCTYPE html>
2  <!-- Declaração do tipo de documento (DOCTYPE),
3      informando ao navegador que esta é
4      uma página HTML5. --&gt;
5
6  &lt;html lang="pt-br"&gt;
7  <!-- Tag de abertura do HTML com o atributo 'lang',
8      especificando que o idioma principal da
9      página é o português do Brasil. --&gt;
10
11 &lt;head&gt;
12  <!-- Tag que contém metadados (informações que
13      descrevem a página) e links para scripts e
14      folhas de estilo. --&gt;
15
16  &lt;meta charset="UTF-8"&gt;
17  <!-- Define o conjunto de caracteres UTF-8 para o
18      documento HTML, suportando uma ampla gama
19      de caracteres de vários idiomas. --&gt;
20
21  &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;
22  <!-- Define a área visível da página web de acordo com o
23      dispositivo usado para visualizá-la, assegurando
24      responsividade. 'initial-scale=1.0' configura o
25      zoom inicial quando a página é carregada pela
26      primeira vez. --&gt;
27
28  &lt;title&gt;Jogo da Forca&lt;/title&gt;
29  <!-- Define o título da página, que é exibido na
30      aba do navegador. --&gt;</pre>
```
- Bottom Status Bar:** Shows file status (0), line (Ln 1, Col 1), spaces (Spaces: 8), encoding (UTF-8), line endings (CRLF), file type (HTML), and live preview (Go Live).

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ..., Back, Forward, Search bar (P30 - Jogo da Forca), Minimize, Maximize, Close.
- Sidebar (EXPLORER):** Shows the project structure:
 - P30 - JOGO DA FORCA
 - index.html
 - p29_jogo_da_forca.png
 - script.js
 - styles.css
- Code Editor:** The current file is index.html, showing the following code:

```
31 <link rel="stylesheet" href="styles.css">
32     <!-- Link para uma folha de estilos externa chamada 'styles.css',
33         que contém as regras de estilização para a página. --&gt;
34
35
36 &lt;/head&gt;
37     <!-- Fecha a tag 'head', indicando o fim da seção de
38         metadados e links. --&gt;
39
40
41 &lt;body&gt;
42     <!-- Tag 'body' que contém todo o conteúdo visível
43         da página. --&gt;
44
45     &lt;div class="container"&gt;
46         <!-- Um contêiner 'div' com a classe 'container' que
47             serve como um bloco de construção para
48             agrupar e estilizar conteúdos relacionados. --&gt;
49
50     &lt;h1&gt;Jogo da Forca&lt;/h1&gt;
51         <!-- Cabeçalho principal 'h1' que fornece o título ou o
52             tópico principal da página: 'Jogo da Forca'. --&gt;
53
54     &lt;div class="pontuacao"&gt;
55         <!-- Divisão com a classe 'pontuacao', usada para
56             exibir a pontuação do jogador. --&gt;
57
58     &lt;p&gt;Pontuação: &lt;span id="pontuacao"&gt;0&lt;/span&gt;&lt;/p&gt;
59         <!-- Parágrafo que mostra a pontuação do jogador.
60             O valor inicial é mostrado dentro de um 'span'</pre>
```

File Edit Selection View Go Run ... ⏪ ⏩ 🔎 P30 - Jogo da Forca

EXPLORER ...

P30 - JOGO DA FORCA

- index.html
- p29_jogo_da_forca.png
- script.js
- styles.css

index.html

```
61 | com o ID 'pontuacao', que pode ser facilmente
62 | acessado e atualizado por scripts. -->
63 |
64 </div>
65 <!-- Fecha a 'div' de pontuação. --&gt;
66
67 &lt;div class="forca-container"&gt;
68   <!-- 'div' com a classe 'forca-container', usada para
69     conter a representação visual do jogo
70     da forca. --&gt;
71
72   &lt;div id="forca"&gt;
73     <!-- 'div' com o ID 'forca', que atua como o espaço
74       principal onde as partes do corpo da
75       forca serão exibidas. --&gt;
76
77     &lt;div id="corda" class="parte"&gt;&lt;/div&gt;
78     <!-- 'div' representando a corda no jogo da
79       forca, com a classe 'parte' para
80       aplicação de estilos específicos. --&gt;
81
82     &lt;div id="cabeça" class="parte"&gt;&lt;/div&gt;
83     <!-- 'div' representando a cabeça na forca,
84       também com a classe 'parte'. --&gt;
85
86     &lt;div id="corpo" class="parte"&gt;&lt;/div&gt;
87     <!-- 'div' para o corpo da forca. --&gt;
88
89     &lt;div id="braçoEsquerdo" class="parte"&gt;&lt;/div&gt;
90     <!-- 'div' para o braço esquerdo da figura da forca. --&gt;</pre>

Ln 61, Col 4 Spaces: 8 UTF-8 CRLF HTML ⏹ Go Live


```

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Icons:** Explorer, Find, Open, Save, Help, Outline, Timeline.
- Left Sidebar (EXPLORER):** P30 - JOGO DA FORCA (expanded) containing index.html, p29_jogo_da_forca.png, script.js, and styles.css.
- Central Area:** Title: index.html X
- Code Structure:** Shows the HTML structure of the hangman game page.
- Code Content:** The code includes comments explaining the purpose of various div elements (braçoDireito, pernaEsquerda, pernaDireita) and sections (palavraSecreta, letras).
- Line Numbers:** 91 to 120.

```
<div id="braçoDireito" class="parte"></div>
<!-- 'div' para o braço direito da figura da forca. --&gt;

&lt;div id="pernaEsquerda" class="parte"&gt;&lt;/div&gt;
<!-- 'div' para a perna esquerda da figura da forca. --&gt;

&lt;div id="pernaDireita" class="parte"&gt;&lt;/div&gt;
<!-- 'div' para a perna direita da figura da forca. --&gt;

&lt;/div&gt;
<!-- Fecha a 'div' de forca. --&gt;

&lt;/div&gt;
<!-- Fecha a 'div' de forca-container. --&gt;

&lt;p&gt;Adivinhe a palavra:&lt;/p&gt;
<!-- Parágrafo que fornece instruções ao jogador
    para adivinhar a palavra. --&gt;

&lt;div id="palavraSecreta"&gt;&lt;/div&gt;
<!-- 'div' vazio com o ID 'palavraSecreta', destinado a
    exibir as letras da palavra secreta à medida
    que são adivinhadas ou inicialmente com
    sublinhados. --&gt;

&lt;div id="letras"&gt;
<!-- 'div' com o ID 'letras', usado para conter e
    organizar a área onde o jogador pode escolher
    letras para adivinhar a palavra secreta. --&gt;</pre>
```

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Explorer:** Shows the project structure under P30 - JOGO DA FORCA, including index.html, p29_jogo_da_forca.png, script.js, and styles.css.
- Code Editor:** The current file is index.html, line 121. The code is as follows:

```
<p>Escolha uma letra:</p>
<!-- Parágrafo que fornece instruções ao jogador
para escolher uma letra do alfabeto abaixo. --&gt;

&lt;div id="alfabeto"&gt;&lt;/div&gt;
<!-- 'div' com o ID 'alfabeto', onde os botões ou
links representando cada letra do
alfabeto serão gerados dinamicamente
via JavaScript. --&gt;

&lt;/div&gt;
<!-- Fecha a 'div' de letras. --&gt;

&lt;button class="btn-reiniciar" onclick="resetGame()"&gt;Reiniciar Jogo&lt;/button&gt;
<!-- Botão com a classe 'btn-reiniciar' que, quando
clicado, aciona a função 'resetGame()'
definida no JavaScript para reiniciar
o jogo. --&gt;

&lt;p id="mensagemFinal"&gt;&lt;/p&gt;
<!-- Parágrafo com o ID 'mensagemFinal', usado para
exibir mensagens finais do jogo, como
"Você venceu!" ou "Você perdeu!". --&gt;

&lt;/div&gt;
<!-- Fecha a 'div' container que agrupa todos os
elementos relacionados ao jogo. --&gt;

&lt;script src="script.js"&gt;&lt;/script&gt;</pre>

The code includes comments explaining the purpose of each section: instructions for the player to choose a letter, a dynamic alphabet container, a button to restart the game, and a final message paragraph.


```

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists files in the current project: `index.html`, `p29_jogo_da_forca.png`, `script.js`, and `styles.css`. The main area displays the content of `index.html`.

```
<!-- Inclui o arquivo JavaScript 'script.js', que
     contém a lógica do jogo, manipulações do DOM e
     interações do usuário. -->
</body>
<!-- Fecha a tag 'body', que contém todo o
     conteúdo visível da página. -->
</html>
<!-- Fecha a tag 'html', que delimita o
     fim do documento HTML. -->
```

The code editor includes standard UI elements like file navigation (`File`, `Edit`, `Selection`, `View`, `Go`, `Run`), search, and zoom controls at the top. The bottom status bar shows file information: `index.html`, `Ln 151, Col 3`, `Spaces: 8`, `UTF-8`, `CRLF`, `HTML`, `Go Live`, and a file count indicator.

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Icons:** Explorer, Search, Find, Open, Save, Help, Outline, Timeline.
- Left Sidebar (EXPLORER):** P30 - JOGO DA FORCA folder containing index.html, p29_jogo_da_forca.png, script.js, and styles.css.
- Active File:** styles.css
- Code Content:** CSS rules for a container class.

```
margin: 0;
/* Remove as margens padrão do body, garantindo que
   não haja espaço extra em torno dos elementos
   internos no início da página. */

}

.container {
    text-align: center;
    /* Alinha o texto dentro dos elementos com a
       classe 'container' ao centro. */

    background-color: #ffffff;
    /* Define a cor de fundo dos elementos com a
       classe 'container' para branco. */

    padding: 20px;
    /* Aplica um preenchimento interno de 20 pixels em
       todos os lados dos elementos com a
       classe 'container', aumentando o espaço
       entre o conteúdo interno e as bordas
       do container. */

    border-radius: 10px;
    /* Arredonda os cantos dos elementos com a
       classe 'container', dando-lhes um raio
       de 10 pixels. */
```

Line numbers 31 through 60 are visible on the left side of the code editor.

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Explorer:** Shows a project structure under P30 - JOGO DA FORCA, including index.html, p29_jogo_da_forca.png, script.js, and styles.css.
- Code Editor:** The file styles.css is open, containing the following CSS code:

```
index.html # styles.css < .container
# styles.css > .container
61    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
62    /* Aplica uma sombra ao redor dos elementos com a
63       classe 'container'. A sombra é suave (10% de
64       opacidade) e se espalha 10 pixels em todas
65       as direções. */
66
67    width: 90%;
68    /* Define a largura dos elementos com a classe 'container'
69       para ser 90% da largura do seu elemento pai, permitindo
70       uma certa flexibilidade em diferentes tamanhos de tela. */
71
72    max-width: 600px;
73    /* Assegura que a largura dos elementos com a
74       classe 'container' não exceda 600 pixels, mantendo a
75       consistência do layout em telas maiores. */
76
77 }
78
79
80 h1 {
81
82    color: #333;
83    /* Define a cor do texto de todos os elementos <h1>
84       para um cinza escuro (#333333), oferecendo boa
85       legibilidade e um design sóbrio. */
86
87 }
88
89 .pontuacao {
```

The code editor has a light theme with syntax highlighting for CSS. The styles.css file is currently selected in the sidebar.

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Icons:** Explorer, Search, Open, Help, Outline, Timeline.
- Left Sidebar (EXPLORER):** P30 - JOGO DA FORCA (index.html, p29_jogo_da_forca.png, script.js, # styles.css).
- Active File:** # styles.css
- Code Content:**

```
91     font-size: 20px;
92     /* Ajusta o tamanho da fonte para 20 pixels nos
93      elementos com a classe "pontuacao", tornando o
94      texto suficientemente grande para fácil leitura. */
95
96     margin-bottom: 10px;
97     /* Aplica uma margem inferior de 10 pixels para separar
98      visualmente a pontuação dos elementos que a seguem. */
99
100 }
101
102 .forca-container {
103
104     display: flex;
105     /* Estabelece um contexto de flexbox para o container do
106      jogo da forca, permitindo um layout flexível dos
107      itens internos. */
108
109     justify-content: center;
110     /* Centraliza horizontalmente os itens dentro
111      do "forca-container", alinhando-os ao
112      centro do container. */
113
114     margin-bottom: 20px;
115     /* Adiciona uma margem inferior de 20 pixels para
116      criar espaço entre o "forca-container" e
117      outros elementos abaixo dele. */
118
119 }
120
```

- Bottom Status Bar:** < 0 △ 0 ⌂ 0 ⌂ 0 Ln 91, Col 5 Spaces: 8 UTF-8 CRLF CSS ⌂ Go Live

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ..., < >, P30 - Jogo da Forca, window control buttons.
- Left Sidebar (EXPLORER):** Shows project files: index.html, p29_jogo_da_forca.png, script.js, and styles.css. styles.css is selected.
- Top Status Bar:** index.html, # styles.css X, # styles.css > #forca.
- Code Editor:** Displays the following CSS code for a hangman game (P30 - JOGO DA FORCA).

```
121 #forca {  
122     position: relative;  
123     /* Define a posição do elemento com ID "forca"  
124      como relativa, o que permite que os elementos  
125      filhos posicionados absolutamente se alinhem  
126      em relação a ele. */  
127  
128     width: 200px;  
129     /* Estabelece uma largura fixa de 200 pixels  
130      para o elemento "forca". */  
131  
132     height: 250px;  
133     /* Estabelece uma altura fixa de 250 pixels para o  
134      elemento "forca", definindo o espaço suficiente  
135      para desenhar a forca e as partes do corpo. */  
136  
137 }  
138 .parte {  
139     display: none;  
140     /* Inicialmente oculta os elementos com a  
141      classe "parte", que representam as diferentes  
142      partes do corpo no jogo da forca. */  
143  
144     position: absolute;  
145     /* Posiciona absolutamente as partes da forca,  
146      permitindo que sejam colocadas em locais  
147      específicos dentro do elemento "forca". */  
148  
149 }  
150 }
```
- Bottom Status Bar:** < > 0 △ 0 ⌂ 0, Ln 121, Col 1, Spaces: 8, UTF-8, CRLF, CSS, Go Live, Bell icon.

A screenshot of a code editor interface, likely Microsoft Visual Studio Code, displaying a file named `# styles.css`. The code is written in CSS and defines styles for a game element. The editor has a dark theme with syntax highlighting. The left sidebar shows project files: `index.html`, `p29_jogo_da_forca.png`, `script.js`, and `# styles.css`. The bottom left shows icons for Explorer, Search, Problems, and Timeline.

```
background-color: black;
/* Define a cor de fundo das partes da forca como
   preto, destacando-as contra o fundo mais claro. */

}

#corda {
    width: 2px;
    /* Define a largura da corda para 2 pixels,
       criando uma linha fina e vertical. */

    height: 40px;
    /* Define a altura da corda para 40 pixels,
       suficiente para simular uma corda pendurada. */

    top: 0;
    /* Posiciona a parte superior da corda no
       topo do elemento "forca". */

    left: 50%;
    /* Posiciona a corda no centro horizontal do
       elemento "forca". */

    transform: translateX(-50%);
    /* Ajusta a posição horizontal da corda para que
       sua linha central fique exatamente no meio,
       corrigindo o posicionamento pelo deslocamento
       da própria largura. */
```

A screenshot of a code editor interface, likely Microsoft Visual Studio Code, displaying a file named `styles.css`. The editor shows the following CSS code:

```
background-color: brown;
/* Define a cor da corda para marrom, simulando a
   cor de uma corda real. */

}

#cabeça {
    width: 30px;
    /* Define a largura da cabeça para 30 pixels. */

    height: 30px;
    /* Define a altura da cabeça para 30 pixels,
       criando uma forma circular perfeita quando
       combinada com a largura igual. */

    border-radius: 50%;
    /* Configura o raio da borda para 50%, o que
       transforma a div em um círculo completo. */

    top: 40px;
    /* Posiciona a cabeça 40 pixels abaixo do topo
       do elemento pai, que é a forca. */

    left: 50%;
    /* Posiciona a cabeça horizontalmente no
       centro do elemento pai. */

    transform: translateX(-50%);
```

The code defines a class `#corda` with a background color of brown. It then defines a class `#cabeça` with a width and height of 30px, a border-radius of 50% (making it a circle), a top position of 40px relative to its parent, a left position of 50% (centering it horizontally), and a transform of `translateX(-50%)` to position it to the left of the center.

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Icons:** Explorer, Search, Open, Find, Outline, Timeline.
- Left Sidebar (EXPLORER):** P30 - JOGO DA FORCA (index.html, p29_jogo_da_forca.png, script.js, styles.css).
- Active File:** styles.css
- Code Content:**

```
/* Ajusta a posição da cabeça movendo-a 50% para
   a esquerda de sua própria largura,
   centralizando-a perfeitamente. */
}

#corpo {
    width: 10px;
    /* Define a largura do corpo para 10 pixels,
       criando uma barra vertical estreita. */

    height: 60px;
    /* Define a altura do corpo para 60 pixels,
       estendendo-se verticalmente abaixo da cabeça. */

    top: 70px;
    /* Posiciona o corpo 70 pixels abaixo do topo
       do elemento pai, começando logo abaixo da cabeça. */

    left: 50%;
    /* Posiciona o corpo horizontalmente no
       centro do elemento pai. */

    transform: translateX(-50%);
    /* Ajusta a posição do corpo movendo-o 50% para
       a esquerda de sua própria largura,
       centralizando-o perfeitamente. */

}
```

- Bottom Status Bar:** < 0 △ 0 ⌂ 0
- Bottom Right Icons:** 🔍 Ln 211, Col 8 Spaces: 8 UTF-8 CRLF CSS ⚡ Go Live 🎙

A screenshot of a code editor interface, likely Microsoft Visual Studio Code, displaying a file named `styles.css`. The code is written in Portuguese and defines styles for a hangman game. The editor has a dark theme with light-colored syntax highlighting. The left sidebar shows project files: `index.html`, `p29_jogo_da_forca.png`, `script.js`, and `styles.css`. The right sidebar shows navigation links for `OUTLINE` and `TIMELINE`.

```
241 #braçoEsquerdo {  
242     width: 10px;  
243     /* Define a largura do braço esquerdo para 10 pixels. */  
244     height: 50px;  
245     /* Define a altura do braço esquerdo para 50 pixels. */  
246     top: 70px;  
247     /* Posiciona o braço esquerdo 70 pixels abaixo do  
248      topo do elemento pai, no mesmo nível que o  
249      topo do corpo. */  
250     left: calc(50% - 25px);  
251     /* Posiciona o braço esquerdo para a esquerda do  
252      centro do corpo usando uma operação  
253      de cálculo. */  
254     transform: rotate(-45deg);  
255     /* Rotaciona o braço esquerdo 45 graus para  
256      cima, criando um ângulo ascendente  
257      para a esquerda. */  
258 }  
259 #braçoDireito {  
260     width: 10px;  
261     /* Define a largura do braço direito para 10 pixels. */  
262 }
```

A screenshot of a code editor interface, likely Microsoft Visual Studio Code, displaying a file named `styles.css`. The editor shows the following CSS code:

```
height: 50px;
/* Define a altura do braço direito para 50 pixels. */

top: 70px;
/* Posiciona o braço direito 70 pixels abaixo do topo do elemento pai, alinhado com o braço esquerdo. */

left: calc(50% + 15px);
/* Posiciona o braço direito para a direita do centro do corpo usando uma operação de cálculo. */

transform: rotate(45deg);
/* Rotaciona o braço direito 45 graus para cima, criando um ângulo ascendente para a direita. */

}

#pernaEsquerda {
    width: 10px;
    /* Define a largura da perna esquerda para 10 pixels. */

    height: 50px;
    /* Define a altura da perna esquerda para 50 pixels. */

    top: 123px;
```

The code defines styles for a right arm and a left leg of a stick figure. The right arm is positioned 70 pixels below its parent element's top, rotated 45 degrees, and has a height of 50 pixels. The left leg is positioned 123 pixels below its parent element's top, has a width of 10 pixels, and a height of 50 pixels.

A screenshot of a code editor interface, likely Microsoft Visual Studio Code, displaying CSS code for a game. The title bar reads "P30 - Jogo da Forca".

The Explorer sidebar shows a project structure with files: index.html, p29_jogo_da_forca.png, script.js, and styles.css.

The current file is styles.css, containing the following CSS:

```
/* Posiciona a perna esquerda 123 pixels abaixo do topo do elemento pai, abaixo do corpo. */
#pernaEsquerda {
    left: calc(50% - 20px);
    /* Posiciona a perna esquerda para a esquerda do centro do corpo usando uma operação de cálculo. */
    transform: rotate(45deg);
    /* Rotaciona a perna esquerda 45 graus para cima, criando um ângulo ascendente para a direita, oposto ao braço esquerdo. */
}

#pernaDireita {
    width: 10px;
    /* Define a largura da perna direita para 10 pixels, mantendo a consistência com as outras partes do corpo. */
    height: 50px;
    /* Define a altura da perna direita para 50 pixels, garantindo que seja longa o suficiente para ser visualmente equilibrada com o resto do corpo. */
    top: 123px;
    /* Posiciona a perna direita a 123 pixels do topo do topo do elemento pai, acima do corpo. */
}
```

A screenshot of a code editor interface, likely Visual Studio Code, displaying CSS code for a game. The title bar reads "P30 - Jogo da Forca".

The Explorer sidebar shows the project structure:

- P30 - JOGO DA FORCA
 - index.html
 - p29_jogo_da_forca.png
 - script.js
 - # styles.css

The current file is "# styles.css".

The code in the editor is:

```
331 # pernaDireita {  
332     |     elemento pai, alinhando-a verticalmente com a  
333     |     perna esquerda. /*  
334     left: calc(50% + 10px);  
335     /* Posiciona a perna direita um pouco à direita do  
336     |     centro do corpo. O 'calc' é usado para  
337     |     adicionar 10 pixels ao deslocamento central,  
338     |     ajustando a posição horizontalmente. */  
339  
340     transform: rotate(-45deg);  
341     /* Rotaciona a perna direita 45 graus para cima e  
342     |     para a esquerda, criando um ângulo que é  
343     |     simetricamente oposto ao da perna esquerda. */  
344 }  
345  
346  
347  
348 #palavraSecreta {  
349  
350     font-size: 24px;  
351     /* Define o tamanho da fonte para 24 pixels, tornando as  
352     |     letras da palavra secreta grandes e fáceis de ler. */  
353  
354     letter-spacing: 15px;  
355     /* Aumenta o espaçamento entre as letras da palavra secreta  
356     |     para 15 pixels, ajudando a separar visualmente cada  
357     |     caractere sublinhado ou revelado. */  
358  
359     margin-bottom: 20px;  
360     /* Adiciona uma margem inferior de 20 pixels, criando
```

A screenshot of a code editor interface, likely Microsoft Visual Studio Code, displaying a CSS file for a Hangman game. The editor has a dark theme with light-colored syntax highlighting.

The left sidebar shows the project structure under "P30 - JOGO DA FORCA", including files "index.html", "script.js", and "# styles.css".

The top navigation bar includes File, Edit, Selection, View, Go, Run, and other standard options.

The search bar at the top center contains the text "P30 - Jogo da Forca".

The main code editor area shows the following CSS code:

```
# palavraSecreta {  
    margin: 10px;  
    /* Define uma margem de 10 pixels entre a palavra secreta e o resto do conteúdo */  
}  
  
.alfabeto {  
    font-weight: bold;  
    /* Deixa a palavra secreta em negrito para destaque */  
}  
  
.alfabeto button {  
    margin: 5px;  
    /* Define uma margem de 5 pixels entre cada botão do alfabeto */  
    padding: 10px;  
    /* Aplica um preenchimento interno de 10 pixels em cada botão, aumentando a área de clique */  
    background-color: #007BFF;  
    /* Define a cor de fundo dos botões para um azul brilhante (#007BFF), tornando-os visualmente atraentes */  
    color: white;  
    /* Define a cor do texto dentro dos botões para branco, garantindo contraste alto com o fundo azul */  
}
```

A screenshot of a code editor interface, likely Visual Studio Code, displaying CSS code for a game. The title bar reads "P30 - Jogo da Forca".

The left sidebar shows the project structure:

- EXPLORER
- P30 - JOGO DA FORCA
 - index.html
 - p29_jogo_da_forca.png
 - script.js
 - # styles.css

The right pane displays the contents of the "# styles.css" file. The code defines styles for an element with the ID "alfabeto button".

```
391 # alfabeto button {  
392     border: none;  
393     /* Remove qualquer borda padrão dos botões, contribuindo  
394      para um design mais limpo e moderno. */  
395  
396     border-radius: 5px;  
397     /* Arredonda as bordas dos botões com um raio de  
398      5 pixels, suavizando sua aparência geral. */  
399  
400     cursor: pointer;  
401     /* Muda o cursor para um ponteiro quando o usuário  
402      passa o mouse sobre os botões, indicando  
403      que eles são interativos. */  
404  
405     font-size: 16px;  
406     /* Define o tamanho da fonte dos botões para 16  
407      pixels, garantindo que o texto seja  
408      suficientemente grande para ser lido  
409      facilmente. */  
410  
411 }  
412  
413  
414 #alfabeto button:disabled {  
415  
416     background-color: #ccc;  
417     /* Define a cor de fundo dos botões do alfabeto que  
418      estão desabilitados para cinza claro (#ccc).  
419      Isso indica visualmente que o botão não está  
420      mais ativo ou interativo. */  
421 }
```

The status bar at the bottom shows: < 0 △ 0 ⌂ 0 Ln 391, Col 1 Spaces: 8 UTF-8 CRLF CSS ⌂ Go Live ⌂

A screenshot of a code editor interface, likely Visual Studio Code, displaying a file named `styles.css`. The editor shows the following CSS code:

```
# alfabeto button:disabled {  
    cursor: not-allowed;  
    /* Muda o cursor para o ícone 'não permitido' quando o  
       usuário passa o mouse sobre um botão desabilitado,  
       reforçando que o botão não pode ser clicado ou  
       utilizado. */  
}  
  
.btn-reiniciar {  
    margin-top: 20px;  
    /* Adiciona uma margem superior de 20 pixels,  
       separando o botão dos elementos acima  
       dele no layout. */  
  
    padding: 10px 20px;  
    /* Define o preenchimento interno do botão,  
       com 10 pixels na vertical e 20 pixels na  
       horizontal, aumentando a área de clique e  
       conforto visual. */  
  
    background-color: #28a745;  
    /* Estabelece a cor de fundo do botão para um  
       verde vibrante (#28a745), tornando-o  
       facilmente identificável como um botão  
       de ação positiva ou de início. */  
  
    color: white;  
    /* Define a cor do texto dentro do botão para
```

The code editor includes a sidebar with icons for Explorer, Search, Open, and Outline, and a bottom bar with navigation and status indicators.

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Icons:** Explorer, Search, Find, Open, Close, Help, Outline, Timeline.
- Project Explorer:** P30 - JOGO DA FORCA (index.html, p29_jogo_da_forca.png, script.js, styles.css).
- Active File:** styles.css (Line 451 to Line 480).
- Code Content:**

```
451     branco, garantindo um contraste alto com o
452         fundo verde para melhor legibilidade. */
453
454     border: none;
455     /* Remove qualquer borda padrão do botão,
456        mantendo o design limpo e moderno. */
457
458     border-radius: 5px;
459     /* Arredonda as bordas do botão com um raio de 5
460        pixels, suavizando a aparência geral e
461        tornando o design mais amigável. */
462
463     cursor: pointer;
464     /* Altera o cursor para um ponteiro quando o
465        usuário passa o mouse sobre o botão,
466        indicando que é clicável. */
467
468     font-size: 16px;
469     /* Ajusta o tamanho da fonte para 16 pixels,
470        fazendo com que o texto dentro do botão
471        seja grande o suficiente para ser lido
472        facilmente. */
473
474 }
475
476 #mensagemFinal {
477
478     font-size: 24px;
479     /* Define o tamanho da fonte para 24 pixels, tornando o
480        texto grande e proeminente, adequado para mensagens
```

- Status Bar:** Line 451, Col 2, Spaces: 8, UTF-8, CRLF, CSS, Go Live, Bell icon.

A screenshot of a code editor interface, likely Microsoft Visual Studio Code, displaying a CSS file named `styles.css` for a project titled "P30 - JOGO DA FORCA".

The code editor shows the following CSS rules:

```
481 #mensagemFinal { finais importantes. /*  
482 color: green;  
483 /* Define a cor do texto para verde, geralmente associado a  
484 | mensagens de sucesso ou conclusão positiva. */  
485 margin-top: 20px;  
486 /* Adiciona uma margem superior de 20 pixels,  
487 | proporcionando espaço entre a mensagem final e  
488 | quaisquer elementos acima dela. */  
489 }  
490  
491 }  
492 }
```

The editor includes standard UI elements such as a top bar with File, Edit, Selection, View, Go, Run, and a search bar. On the left is a sidebar with icons for Explorer, Search, Problems, and Outline/Timeline. The bottom right corner shows status information like line number (Ln 481, Col 1), spaces used (Spaces: 8), and file encoding (UTF-8).

The screenshot shows a Microsoft Visual Studio Code (VS Code) window with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Icons:** Explorer, Search, Open, Find, Preview, Outline, Timeline.
- Left Sidebar (EXPLORER):** Shows a folder named "P30 - JOGO DA FORCA" containing "index.html", "p29_jogo_da_forca.png", "script.js" (selected), and "# styles.css".
- Central Area:** A code editor with the following content:

```
const palavras = ['javascript', 'css', 'html', 'navegador', 'programar', 'internet', 'computador', 'teclado', 'mouse', 'monitor'];
/* Declara uma constante 'palavras' e a inicializa com um array de strings. Cada string representa uma palavra que pode ser usada no jogo da forca. */

const pontuacaoKey = 'jogoDaForcaPontuacao';
/* Declara uma constante 'pontuacaoKey' e a define como 'jogoDaForcaPontuacao', que será usada como chave para acessar a pontuação armazenada no localStorage. */

let palavraSecreta = '';
/* Declara uma variável 'palavraSecreta' e inicializa como uma string vazia. Esta variável será usada para armazenar a palavra atual que o jogador deve adivinhar. */

let erros = 0;
/* Declara uma variável 'erros' e inicializa com zero. Esta variável contará o número de erros que o jogador comete ao adivinhar letras incorretas. */

let acertos = 0;
/* Declara uma variável 'acertos' e inicializa com zero. Esta variável contará o número de acertos que o jogador faz ao adivinhar letras corretas. */

let pontuacao = localStorage.getItem(pontuacaoKey) ? parseInt(localStorage.getItem(pontuacaoKey)) : 0;
/* Declara uma variável 'pontuacao' e a inicializa com o valor armazenado no localStorage, convertido para número inteiro. Se não houver valor, inicia em 0.
```

The code is written in JavaScript and defines variables for words, scores, secret word, errors, and correct answers, along with logic for loading the score from local storage.

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ..., < >, P30 - Jogo da Forca, window control buttons.
- Left Sidebar (EXPLORER):** Shows a project structure under P30 - JOGO DA FORCA, including index.html, p29_jogo_da_forca.png, script.js (selected), and styles.css.
- Central Area:** A code editor with tabs for index.html, styles.css, script.js (highlighted in blue), and an X icon.
- Code Content:** The script.js file contains the following code:

```
valor armazenado no localStorage usando a
chave 'pontuacaoKey'. Se não houver valor
armazenado, inicializa com zero.

'parseInt' é usado para converter o valor recuperado,
que é uma string, em um número inteiro. */

let indicePalavra = localStorage.getItem('indicePalavra') ? parseInt(localStorage.getItem('indicePalavra')) : 0;
/* Declara uma variável 'indicePalavra' e a inicializa com o
   valor armazenado no localStorage para 'indicePalavra'.
   Se não houver valor armazenado, inicializa com zero.
   'parseInt' é usado para garantir que o valor seja um
   número inteiro. */

document.getElementById('pontuacao').textContent = pontuacao;
/* Acessa o elemento HTML com o ID 'pontuacao' e define
   seu conteúdo de texto para o valor da
   variável 'pontuacao'.
   Isso atualiza a exibição da pontuação na página
   com o valor atual armazenado em 'pontuacao'. */

function escolherPalavraSecreta() {
    /* Define a função chamada 'escolherPalavraSecreta'.
       Esta função é responsável por selecionar uma
       palavra do array 'palavras' para ser a palavra
       que o jogador deve adivinhar no jogo. */

    if (indicePalavra < palavras.length) {
        /* Verifica se o valor da variável 'indicePalavra' é
           menor que o comprimento do array 'palavras'.
```

Bottom status bar: < > 0 △ 0 ⌂ 0 Q Ln 30, Col 2 Spaces: 7 UTF-8 CRLF {} JavaScript ⌂ Go Live ⌂

File Edit Selection View Go Run ... ← → P30 - Jogo da Forca

EXPLORER ... index.html # styles.css JS script.js X

JS script.js > escolherPalavraSecreta

```
60     Isso é usado para garantir que o índice não ultrapasse o
61         número de palavras disponíveis no array, evitando
62             erros de índice fora dos limites. */
63
64     palavraSecreta = palavras[indicePalavra];
65     /* Se a condição for verdadeira, ou seja, se ainda
66         há palavras no array que não foram usadas
67         com base no índice atual,
68         atribui à variável 'palavraSecreta' a palavra
69         localizada na posição 'indicePalavra'
70         do array 'palavras'.
71         Isso configura 'palavraSecreta' como a
72             próxima palavra a ser adivinhada no jogo. */
73
74 } else {
75     /* Se não houver mais palavras no array dentro do
76         alcance do 'indicePalavra' (ou seja, se 'indicePalavra'
77         for igual ou maior que o comprimento do array),
78         a condição acima retorna falso e esse bloco de
79         código será executado. */
80
81     palavraSecreta = palavras[Math.floor(Math.random() * palavras.length)];
82     /* Seleciona uma palavra aleatoriamente do array 'palavras'.
83         'Math.random()' gera um número aleatório entre 0 e 1 (exclusivo),
84         que é multiplicado pelo comprimento do
85         array 'palavras' para obter um índice aleatório.
86         'Math.floor()' é usado para arredondar o número gerado
87         para o inteiro mais próximo para baixo, garantindo
88         que o índice seja um número inteiro válido
89         dentro do alcance do array.
```

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ..., <-, >, P30 - Jogo da Forca, window control buttons.
- Left Sidebar (EXPLORER):** Shows a project structure under "P30 - JOGO DA FORCA" with files: index.html, styles.css, script.js (selected), and p29_jogo_da_forca.png.
- Central Area:** A code editor with tabs for index.html, styles.css, script.js (highlighted in blue), and a closed tab.
- Code Content:** The script.js file contains the following code in Portuguese:

```
index.html    # styles.css  JS script.js  X
JS script.js > escolherPalavraSecreta

90     A palavra no índice gerado aleatoriamente é então
91         | atribuída à variável 'palavraSecreta'. */
92
93     }
94 }

95
96 function montarPalavraNaTela() {
97     /* Define a função 'montarPalavraNaTela'. Esta função é
98         | responsável por exibir visualmente a
99         | palavra secreta no jogo da forca,
100        | representando cada letra por um
101        | sublinhado inicialmente. */
102
103 const container = document.getElementById('palavraSecreta');
104 /* Acessa o elemento HTML com o ID 'palavraSecreta'.
105     | Este elemento serve como o contêiner no
106     | qual as letras da palavra secreta serão exibidas.
107     | A variável 'container' agora referencia este elemento DOM. */
108
109 container.innerHTML = '';
110 /* Limpa o conteúdo interno do elemento 'container',
111     | removendo quaisquer letras ou sublinhados que
112     | possam ter sido exibidos anteriormente.
113     | Isso é útil para garantir que o contêiner esteja
114     | vazio antes de adicionar a nova representação
115     | da palavra secreta. */
116
117 for (let letra of palavraSecreta) {
118     /* Inicia um loop que percorre cada letra na
119         | string 'palavraSecreta'. A variável 'letra'
```

The code is a function named `montarPalavraNaTela` which retrieves an element with the ID `'palavraSecreta'` from the DOM and sets its `innerHTML` to an empty string. It then loops through each character in the secret word and inserts it into the container element. The code uses inline comments to explain the purpose of each step.

The screenshot shows a code editor interface with the following details:

- File Bar:** File Edit Selection View Go Run ...
- Search Bar:** P30 - Jogo da Forca
- Explorer:** P30 - JOGO DA FORCA (index.html, p29_jogo_da_forca.png, script.js, styles.css)
- Editor:** JS script.js (highlighted)
- Code Content:** A snippet of JavaScript for building a hangman game. It creates a span element, sets its textContent to an underscore, adds a 'letra' class, and appends it to a container.
- Bottom Status Bar:** < > 0 △ 0 ⌂ 0 < > 000 - X
- Bottom Icons:** OUTLINE, TIMELINE, Go Live, and a bell icon.

```
120     receberá, a cada iteração, um caractere da
121     string 'palavraSecreta'. */
122
123     const span = document.createElement('span');
124     /* Cria um novo elemento 'span'. Este elemento será usado
125        para representar uma única letra da palavra
126        secreta no HTML. */
127
128     span.textContent = '_';
129     /* Define o conteúdo de texto do elemento 'span' para
130        um sublinhado ('_').
131        Isso é usado para indicar que a letra ainda não
132        foi adivinhada. */
133
134     span.classList.add('letra');
135     /* Adiciona a classe 'letra' ao elemento 'span'.
136        Esta classe pode ser usada para aplicar
137        estilos específicos, como fonte, cor e espaçamento,
138        e para identificar facilmente esses
139        elementos como parte da palavra secreta no DOM. */
140
141     container.appendChild(span);
142     /* Anexa o elemento 'span' criado ao contêiner
143        'palavraSecreta'. Isso faz com que o
144        sublinhado apareça no documento HTML.
145        Cada 'span' adicionado representa uma letra
146        da palavra secreta. */
147
148 }
149 }
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) window with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Icons:** Explorer, Search, Open, Find, Preview, Outline, Timeline.
- Project Explorer:** P30 - JOGO DA FORCA contains index.html, p29_jogo_da_forca.png, script.js, and styles.css.
- Text Editor:** The script.js file is open, showing the following code:

```
150
151
152 function montarAlfabeto() {
153     /* Define a função chamada 'montarAlfabeto'. Essa
154      função cria botões para cada letra do
155      alfabeto e os exibe na tela, permitindo ao
156      jogador escolher letras para adivinhar a
157      palavra secreta. */
158
159 const alfabeto = 'abcdefghijklmnopqrstuvwxyz';
160 /* Declara uma constante chamada 'alfabeto' e a
161     inicializa com uma string contendo todas as
162     letras do alfabeto em minúsculas. Esta string
163     será usada para iterar e criar um botão
164     para cada letra. */
165
166 const container = document.getElementById('alfabeto');
167 /* Acessa o elemento HTML com o ID 'alfabeto'. Este
168     elemento atua como container para os botões
169     de letras que serão criados.
170     A variável 'container' agora referencia este
171     elemento do DOM. */
172
173 container.innerHTML = '';
174 /* Limpa o conteúdo interno do elemento 'container',
175     garantindo que não haja elementos residuais
176     antes de adicionar os novos botões.
177     Isso é importante para evitar duplicação de botões
178     quando a função é chamada mais de uma vez. */
179
```

The code defines a function `montarAlfabeto()` which creates buttons for each letter of the alphabet and displays them on the screen. It uses a constant `alfabeto` containing all lowercase letters and a variable `container` to hold the HTML element with ID `'alfabeto'`. The function then clears the `innerHTML` of the container to prevent duplicates when called multiple times.

The screenshot shows a Microsoft Visual Studio Code (VS Code) window with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Icons:** Explorer, Search, Open, Find, Split, Outline, Timeline.
- Active File:** script.js
- Content:** A snippet of JavaScript code for a Hangman game. The code uses a forEach loop to iterate over buttons, checking if the current character matches the chosen letter. It updates the span's text content to uppercase if it matches and increments the 'acertos' counter if true. The code is annotated with comments explaining its purpose.

```
index.html styles.css script.js
JS script.js > JS escolherLetra > JS forEach() callback
240     com um loop 'forEach'.
241     'char' representa o caractere atual na iteração
242     e 'index' é o índice desse caractere na
243     palavra. */
244
245     if (char === letra) {
246         /* Verifica se o caractere atual é igual à
247             |     |     letra escolhida pelo jogador. */
248
249         spans[index].textContent = char.toUpperCase();
250         /* Se o jogador acertar a letra, atualiza o
251             |     |     conteúdo de texto do 'span' correspondente
252             |     |     na posição 'index' para mostrar a
253             |     |     letra em maiúscula. */
254
255         acertou = true;
256         /* Atualiza a variável 'acertou' para 'true',
257             |     |     indicando que o jogador acertou pelo
258             |     |     menos uma letra. */
259
260         acertos++;
261         /* Incrementa a variável 'acertos', que conta
262             |     |     quantas letras corretas foram adivinhadas
263             |     |     pelo jogador durante o jogo. */
264
265     }
266 });
267
268
269 document.querySelectorAll('#alfabeto button').forEach(button => {
```

The screenshot shows a code editor interface with the following details:

- File Bar:** File Edit Selection View Go Run ...
- Search Bar:** P30 - Jogo da Forca
- Icons:** Explorer, Search, Find, Open, Save, Help, Outline, Timeline.
- Project Explorer (Left):** P30 - JOGO DA FORCA contains index.html, p29_jogo_da_forca.png, script.js, and styles.css.
- Active File:** script.js
- Code Content:** The code is written in JavaScript and handles the logic for a Hangman game. It iterates over buttons in a container, checks if the button's text matches a chosen letter, disables it if true, and increments error count if false. It also calls a function to show parts of a gallows figure based on error count.

```
index.html    # styles.css  JS script.js  X
JS script.js > escolherLetra > forEach() callback
270     /* Seleciona todos os botões dentro do elemento com
271         ID 'alfabeto' e itera sobre cada botão
272         com um loop 'forEach'. */
273
274     if (button.textContent.toLowerCase() === letra) {
275         /* Verifica se o texto do botão (convertido para
276             minúscula) é igual à letra escolhida pelo jogador. */
277
278         button.disabled = true;
279         /* Desabilita o botão correspondente à letra escolhida,
280             impedindo que a mesma letra seja escolhida
281             novamente. */
282
283     }
284 });
285
286 if (!acertou) {
287     /* Verifica se a variável 'acertou' ainda é 'false', o
288         que significa que o jogador não acertou
289         nenhuma letra da palavra secreta. */
290
291     erros++;
292     /* Incrementa a variável 'erros', que conta quantos
293         erros o jogador cometeu ao adivinhar
294         letras incorretas. */
295
296     mostrarParteDoBoneco(erros);
297     /* Chama a função 'mostrarParteDoBoneco', passando a
298         quantidade atual de erros, que exibe uma
299         nova parte do boneco na forca, indicando
```

- Status Bar:** Ln 270, Col 4 Spaces: 7 UTF-8 CRLF {} JavaScript Go Live

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Icons:** Explorer, Search, Open, Find, Outline, Timeline.
- Project Explorer:** P30 - JOGO DA FORCA (index.html, p29_jogo_da_forca.png, script.js, styles.css).
- Active File:** script.js
- Code Content:** A snippet of JavaScript code for a hangman game. The code includes functions for checking if the game is over and displaying parts of the hangman based on errors made.

```
index.html # styles.css JS script.js X
JS script.js > escolherLetra
300     visualmente um erro. */
301
302 }
303
304     verificarFimDeJogo();
305     /* Chama a função 'verificarFimDeJogo', que verifica
306         se o jogo terminou, seja por o jogador ter
307         adivinhado a palavra completa ou por ter
308         cometido o máximo de erros permitidos. */
309
310 }
311
312
313
314 function mostrarParteDoBoneco(erros) {
315     /* Define a função 'mostrarParteDoBoneco' que é
316         chamada para atualizar a visualização do
317         boneco com base no número de erros
318         cometidos pelo jogador.
319         O parâmetro 'erros' representa o número total de
320         erros que o jogador acumulou até o momento. */
321
322     const partes = ['corda', 'cabeça', 'corpo', 'braçoEsquerdo', 'braçoDireito', 'pernaEsquerda', 'pernaDireita'];
323     /* Declara um array chamado 'partes' contendo strings
324         que correspondem aos IDs dos elementos HTML
325         que representam as partes do boneco da forca.
326         Cada elemento no array é um nome de parte que precisa
327         ser visualizada quando um erro é cometido. */
328
329     if (erros <= partes.length) {
```

The screenshot shows a code editor interface with the following details:

- File Bar:** File Edit Selection View Go Run ...
- Search Bar:** P30 - Jogo da Forca
- Icons:** Explorer, Search, Open, Find, Outline, Timeline.
- Project Explorer:** P30 - JOGO DA FORCA (index.html, p29_jogo_da_forca.png, script.js, styles.css).
- Active File:** script.js
- Code Content:** The code is written in Portuguese and handles the logic for displaying parts of a hangman figure based on the number of errors made. It includes comments explaining the logic of displaying parts and hiding them using the 'display' CSS property.

```
/* Verifica se o número de erros é menor ou igual ao
 * número de partes do boneco que podem ser mostradas.
 * Isso evita tentativas de acessar partes que não
 * existem no array caso o número de erros
 * exceda o número de partes definidas. */

const parte = document.getElementById(partes[erros - 1]);
/* Acessa o elemento DOM correspondente à parte do
 * boneco que deve ser mostrada.
 * 'erros - 1' é usado como índice porque os arrays em
 * JavaScript são indexados a partir de zero,
 * então 'erros - 1' corresponde à próxima parte do
 * boneco a ser revelada de acordo com o
 * número de erros. */

parte.style.display = 'block';
/* Altera a propriedade de estilo 'display' do
 * elemento selecionado para 'block', tornando a
 * parte visível na interface do usuário.
 * Inicialmente, todas as partes são ocultadas com
 * 'display: none', e esta linha as torna visíveis à
 * medida que os erros são cometidos. */

}

}

function verificarFimDeJogo() {
/* Define a função 'verificarFimDeJogo' que é chamada
 * para verificar se o jogo alcançou uma
```

- Status Bar:** Ln 330, Col 6 Spaces: 7 UTF-8 CRLF {} JavaScript Go Live

File Edit Selection View Go Run ... ← → 🔍 P30 - Jogo da Forca

EXPLORER ...

P30 - JOGO DA FORCA

- index.html
- # styles.css
- script.js
- verificarFimDeJogo

360 | condição de término,
361 | seja porque o jogador cometeu erros demais ou
362 | porque adivinhou corretamente todas as
363 | letras da palavra. */
364
365 const mensagem = document.getElementById('mensagemFinal');
366 /* Acessa o elemento HTML com o ID 'mensagemFinal',
367 | que é usado para exibir mensagens ao final do jogo.
368 A variável 'mensagem' agora referencia este
369 | elemento do DOM. */
370
371 if (erros === 7) {
372 | /* Verifica se o número de erros cometidos pelo
373 | jogador é igual a 7, o que significa que o
374 | jogador usou todas as suas tentativas permitidas
375 | sem adivinhar a palavra corretamente. */
376
377 mensagem.textContent = 'Você perdeu!';
378 /* Atualiza o conteúdo de texto do elemento 'mensagem'
379 | para 'Você perdeu!', informando ao jogador
380 | que ele não conseguiu adivinhar a palavra. */
381
382 document.querySelectorAll('#alfabeto button').forEach(button => button.disabled = true);
383 /* Seleciona todos os botões do alfabeto e os
384 | desabilita. Isso é feito para evitar que o
385 | jogador continue tentando adivinhar
386 | após o jogo ter terminado. */
387
388 } else if (acertos === palavraSecreta.length) {
389 | /* Verifica se o número de acertos é igual ao

Ln 360, Col 6 Spaces: 7 UTF-8 CRLF {} JavaScript ⚡ Go Live

The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Toolbar:** Minimize, Maximize, Close, Fullscreen, Refresh, Save, Undo, Redo.
- Left Sidebar (EXPLORER):** Shows the project structure under "P30 - JOGO DA FORCA": index.html, styles.css, script.js, and p29_jogo_da_forca.png. The script.js file is selected.
- Top Tabs:** index.html, # styles.css, JS script.js (highlighted), and X.
- Code Editor:** Displays the content of script.js. The code handles winning conditions, disables buttons, increments score, moves to the next word, and saves progress to localStorage.
- Bottom Status Bar:** Includes icons for file operations, a search bar, and status information: Ln 390, Col 7, Spaces: 7, UTF-8, CRLF, {}, JavaScript, Go Live, and a refresh icon.

```
index.html    # styles.css  JS script.js  X
JS script.js > verificarFimDeJogo

390     | número de letras da palavra secreta, o que
391     | indica que o jogador adivinhou
392     | todas as letras corretamente. */
393
394     mensagem.textContent = 'Você venceu!';
395     /* Atualiza o conteúdo de texto do elemento 'mensagem'
396      para 'Você venceu!', celebrando o sucesso do
397      jogador em adivinhar a palavra completa. */
398
399     document.querySelectorAll('#alfabeto button').forEach(button => button.disabled = true);
400     /* Semelhante ao caso de derrota, desabilita todos os
401      botões do alfabeto para prevenir mais interações
402      após o jogo ter terminado. */
403
404     pontuacao += 1;
405     /* Incrementa a pontuação do jogador em 1, recompensando-o
406      por ter adivinhado a palavra corretamente. */
407
408     indicePalavra++;
409     /* Incrementa o 'indicePalavra' para avançar para a
410      próxima palavra na lista de palavras do jogo
411      na próxima rodada. */
412
413     localStorage.setItem('indicePalavra', indicePalavra);
414     /* Salva o novo valor de 'indicePalavra' no localStorage,
415      garantindo que o progresso do jogador seja
416      mantido entre sessões do navegador. */
417
418 }
419
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) window with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Icons:** Explorer, Search, Problems, Editor, Terminal, User, Settings.
- Left Sidebar (EXPLORER):** Shows a project folder "P30 - JOGO DA FORCA" containing "index.html", "p29_jogo_da_forca.png", "script.js", and "# styles.css".
- Active File:** "script.js" is the active file, indicated by a yellow background and a yellow icon in the sidebar.
- Code Editor:** The code is written in JavaScript and handles the logic for a Hangman game. It includes functions for saving scores to localStorage, updating the UI, and starting a new game.
- Bottom Status Bar:** Shows file count (0), line count (449), column count (5), spaces (7), and encoding (UTF-8). It also shows tabs for "OUTLINE" and "TIMELINE".

```
index.html # styles.css JS script.js X
JS script.js > verificarFimDeJogo
420     localStorage.setItem(pontuacaoKey, pontuacao);
421     /* Salva a pontuação atualizada no localStorage
422      | usando a chave 'pontuacaoKey'. */
423
424     document.getElementById('pontuacao').textContent = pontuacao;
425     /* Atualiza o conteúdo de texto do elemento com ID 'pontuacao'
426      | para refletir a nova pontuação, garantindo que a
427      | interface mostre o valor correto. */
428
429 }
430
431 function resetGame() {
432     /* Define a função 'resetGame', que é responsável por
433      | reiniciar o jogo, redefinindo todas as
434      | variáveis e estados para permitir um novo começo. */
435
436     escolherPalavraSecreta();
437     /* Chama a função 'escolherPalavraSecreta' para selecionar
438      | uma nova palavra secreta do conjunto disponível.
439      | Isso garante que cada novo jogo comece com uma palavra
440      | diferente ou escolhida aleatoriamente. */
441
442     montarPalavraNaTela();
443     /* Chama a função 'montarPalavraNaTela' para atualizar a
444      | visualização da palavra secreta na tela.
445      | Inicialmente, mostra sublinhados onde cada letra da
446      | palavra aparecerá à medida que forem sendo
447      | adivinhadas. */
448
449     montarAlfabeto();
```


The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P30 - Jogo da Forca
- Toolbar:** Standard window control icons (minimize, maximize, close).
- Left Sidebar:**
 - EXPLORER
 - P30 - JOGO DA FORCA
 - index.html
 - p29_jogo_da_forca.png
 - script.js** (selected)
 - styles.css
- Code Editor:** The selected file is 'script.js'. The code is as follows:

```
480 // Inicialização do jogo
481 escolherPalavraSecreta();
482 /* Chama a função 'escolherPalavraSecreta' ao carregar o script
483 para definir a primeira palavra secreta para o jogo inicial. */
484
485 montarPalavraNaTela();
486 /* Chama a função 'montarPalavraNaTela' ao carregar o script para
487 preparar a exibição inicial da palavra secreta
488 com sublinhados. */
489
490 montarAlfabeto();
491 /* Chama a função 'montarAlfabeto' ao carregar o script para criar o
492 teclado virtual que permite ao jogador escolher letras
493 para tentar adivinhar a palavra secreta. */
```
- Bottom Status Bar:** Icons for file operations (New, Open, Save, Find, Replace), status (Ln 480, Col 4, Spaces: 7, UTF-8, CRLF), and language (JavaScript). There are also 'Go Live' and 'Timeline' buttons.

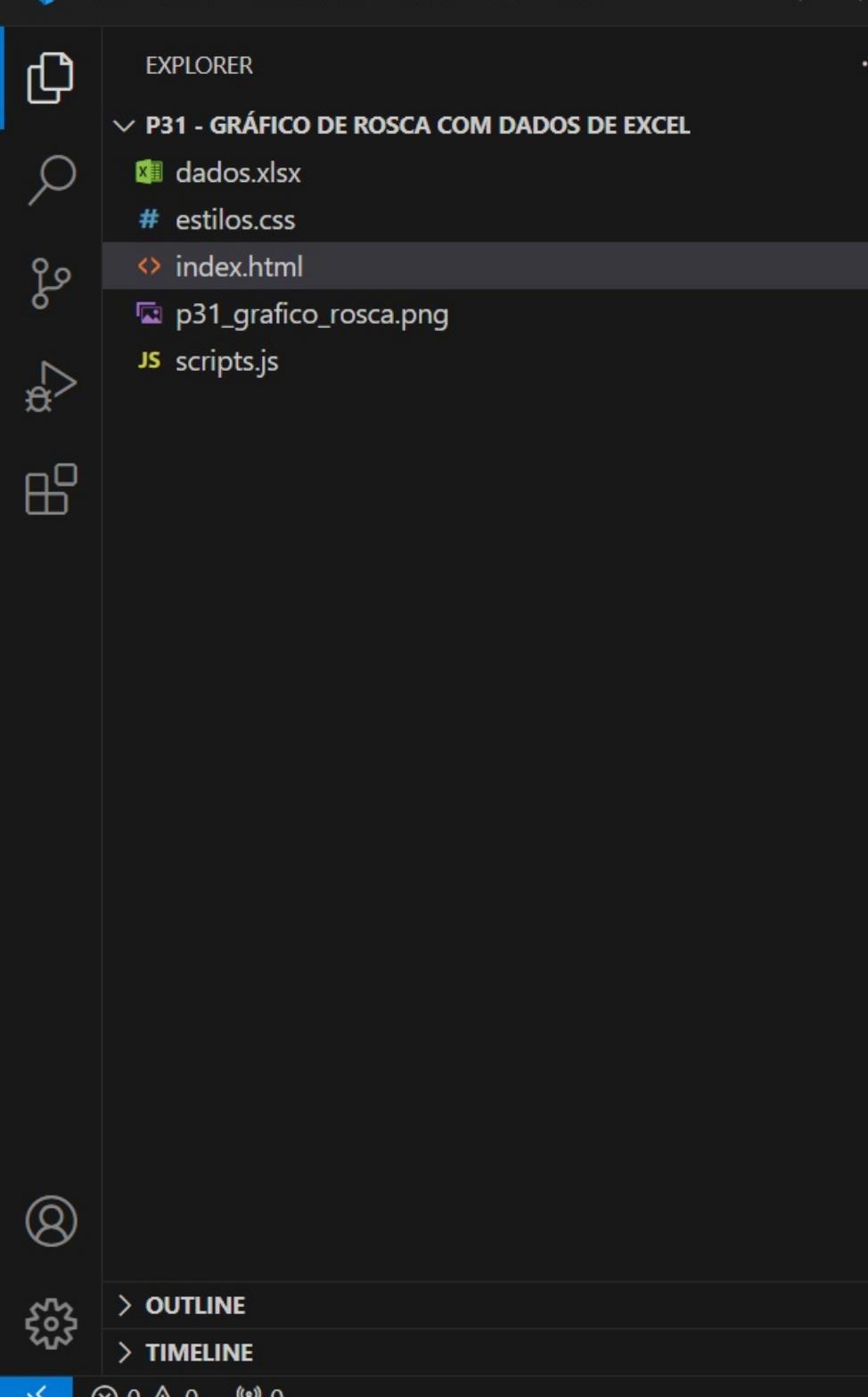
UC: Projetos

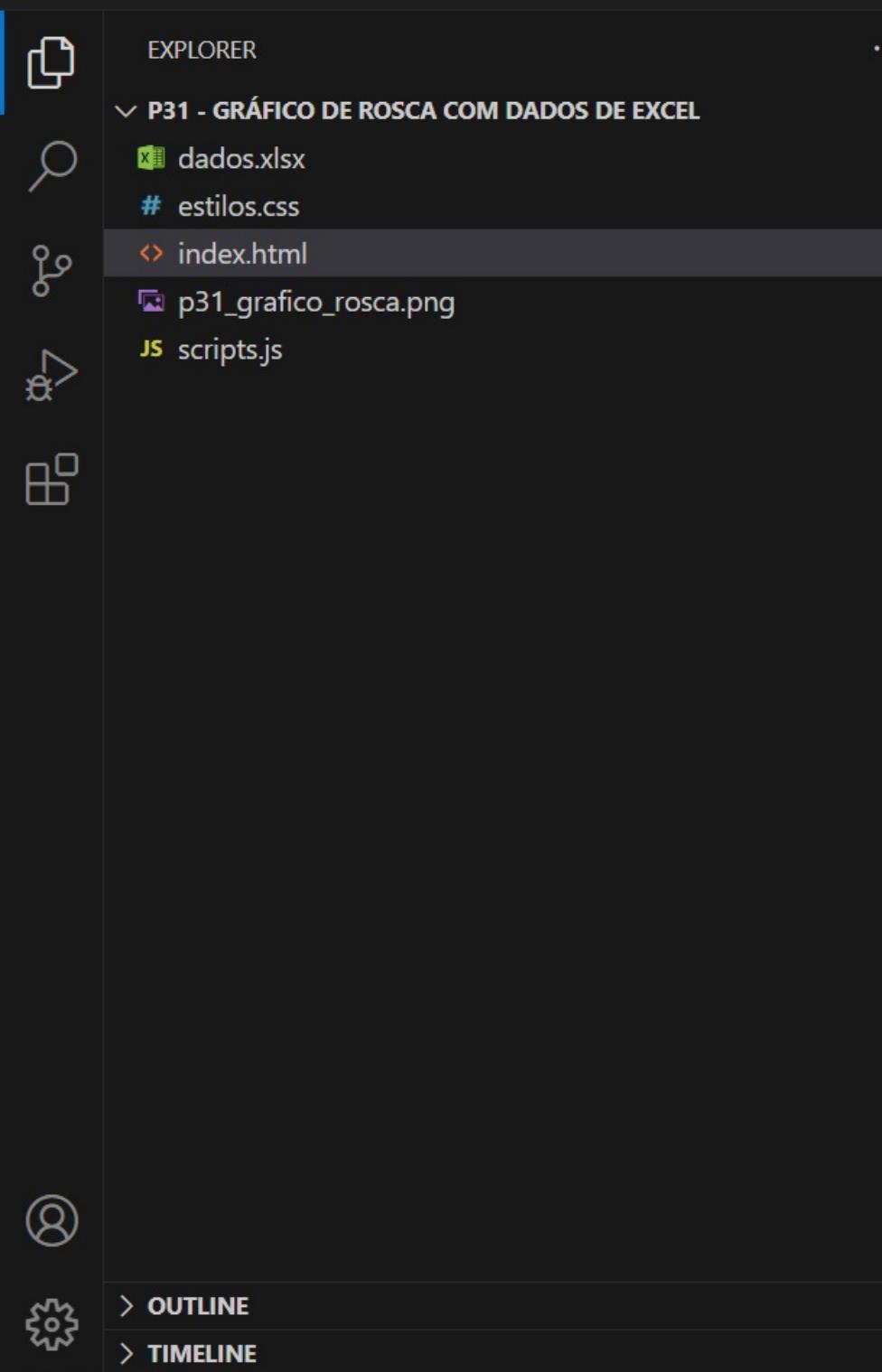
DIAS: 25/10/2024

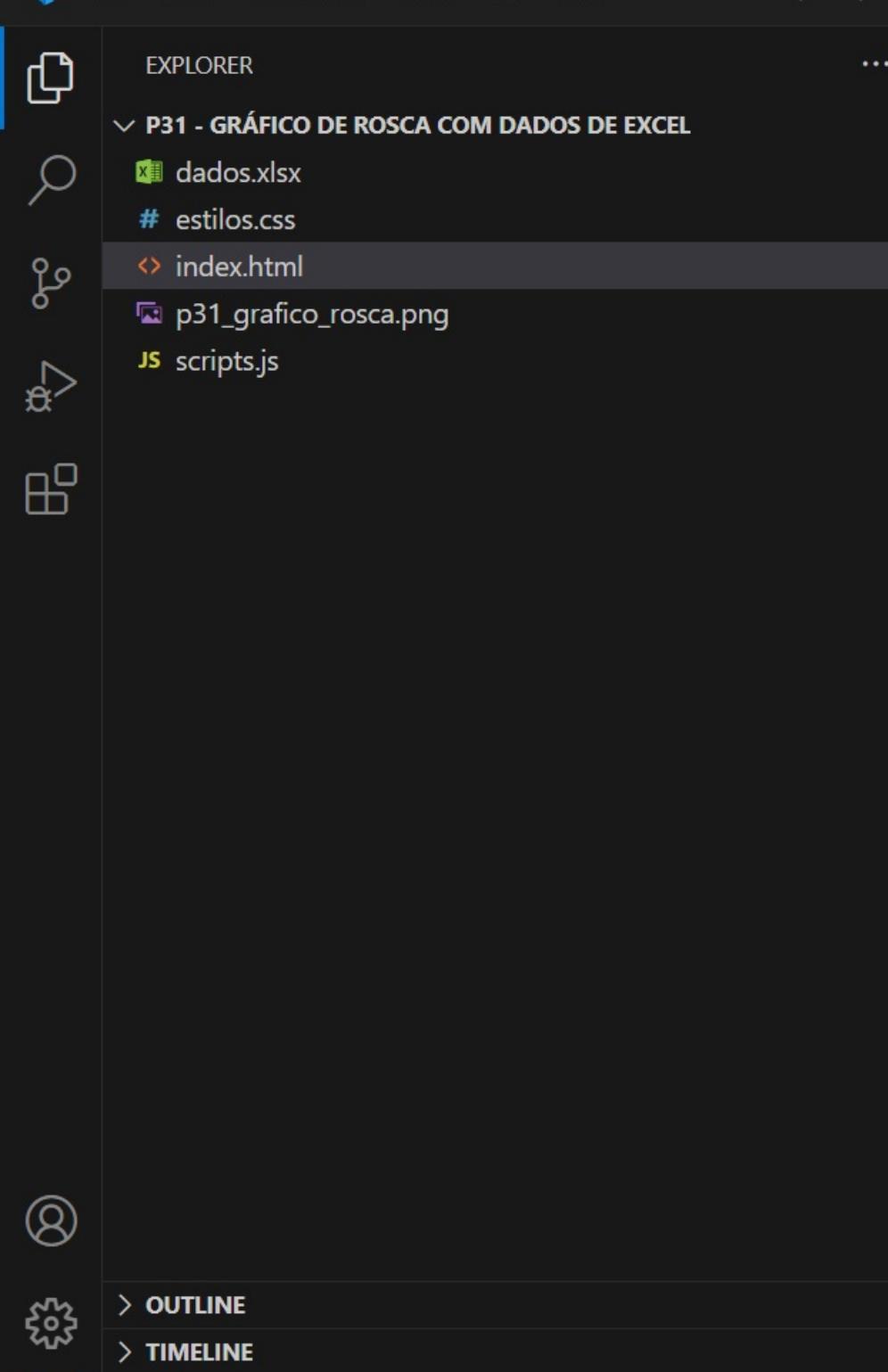
Assunto:

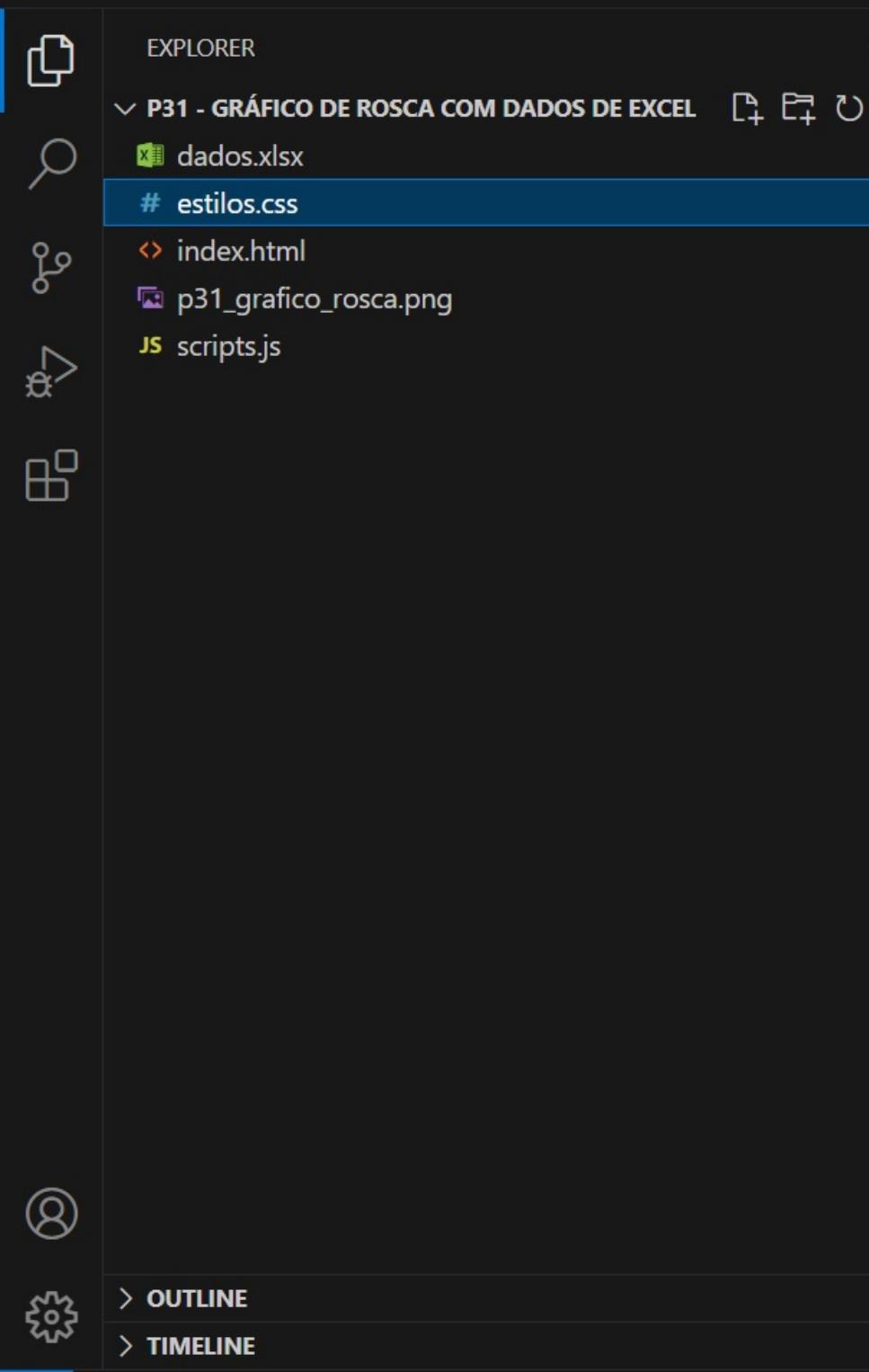
Small Projects

P31 – Gráfico de Rosca com Dados do Excel









File Edit Selection View Go Run ... ← → P31 - Gráfico de Rosca com Dados de Excel

EXPLORER

P31 - GRÁFICO DE ROSCA COM DADOS DE EXCEL

- dados.xlsx
- # estilos.css
- index.html
- p31_grafico_rosca.png
- scripts.js

index.html # estilos.css JS scripts.js

```
# estilos.css > .container
61 border-radius: 10px;
62 /* Aplica um raio de borda de 10 pixels, arredondando os cantos do
63 | container para um visual moderno e suave. */
64
65 box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
66 /* Adiciona uma sombra abaixo do container, utilizando RGBA para uma
67 | cor preta com 10% de opacidade. Isso dá ao container um
68 | efeito de elevação sutil, melhorando a percepção visual
69 | da profundidade. */
70
71 max-width: 800px;
72 /* Define a largura máxima do container para 800 pixels, garantindo
73 | que ele não se torne demasiado largo em monitores maiores, o
74 | que poderia prejudicar a experiência visual e de leitura. */
75
76 width: 100%;
77 /* Faz com que o container tome toda a largura disponível até o máximo de 800
78 | pixels, permitindo que ele seja responsivo e se ajuste a diferentes
79 | tamanhos de tela. */
80
81 }
82
83 h1 {
84
85 margin-bottom: 20px;
86 /* Esta regra define uma margem inferior de 20 pixels para todos os
87 | elementos <h1>. O espaço extra na parte inferior
88 | serve para separar visualmente o título do conteúdo que
89 | segue, como um <canvas> ou outro texto, melhorando a legibilidade
90 | e a estética geral da página ao evitar que os elementos fiquem
```

Ln 61, Col 5 Spaces: 4 UTF-8 CRLF CSS Go Live

File Edit Selection View Go Run ... ← → 🔍 P31 - Gráfico de Rosca com Dados de Excel

EXPLORER

P31 - GRÁFICO DE ROSCA COM DADOS DE EXCEL

- dados.xlsx
- # estilos.css
- index.html
- p31_grafico_rosca.png
- scripts.js

index.html # estilos.css JS scripts.js

estilos.css > h1

```
91         visualmente amontoados. */
92
93     }
94
95     canvas {
96
97         margin-top: 20px;
98         /* Esta regra aplica uma margem superior de 20 pixels a todos os
99            elementos <canvas>. Assim como a margem no <h1>,
100           esta margem cria espaço entre o <canvas> e qualquer conteúdo
101             acima dele, como um título ou outro elemento, garantindo
102               que o gráfico não toque diretamente em outros conteúdos e
103                 oferecendo uma apresentação mais clara e organizada. */
104
105 }
```

...

OUTLINE

TIMELINE

✖ 0 ⚠ 0 ⚡ 0

Ln 91, Col 1 Spaces: 4 UTF-8 CRLF CSS Go Live

EXPLORER
P31 - GRÁFICO DE ROSCA COM DADOS DE EXCEL

- dados.xlsx
- # estilos.css
- index.html
- p31_grafico_rosca.png
- scripts.js

- > OUTLINE
- > TIMELINE

```
index.html # estilos.css JS scripts.js X
JS scripts.js > document.addEventListener('DOMContentLoaded') callback > then() callback
31           informa à função que os dados estão em um array
32           de bytes (ArrayBuffer). */
33
34           // Obtém o nome da primeira planilha do arquivo Excel.
35           var nomeDaPlanilha = workbook.SheetNames[0];
36           /* 'workbook.SheetNames' é um array que contém os nomes de
37           |   |   todas as planilhas no arquivo Excel. '[0]' acessa o nome
38           |   |   da primeira planilha, assumindo que é a planilha de interesse. */
39
40           // Acessa a planilha pelo seu nome para obter os dados.
41           var planilha = workbook.Sheets[nomeDaPlanilha];
42           /* 'workbook.Sheets' é um objeto que contém todas as planilhas
43           |   |   como propriedades, acessíveis pelo nome da planilha. */
44
45           // Converte os dados da planilha em formato JSON.
46           var dadosJson = XLSX.utils.sheet_to_json(planilha);
47           /* 'XLSX.utils.sheet_to_json' converte a planilha especificada em
48           |   |   um array de objetos JSON, onde cada objeto representa
49           |   |   uma linha da planilha, facilitando a manipulação e
50           |   |   visualização dos dados em JavaScript. */
51
52           // Chama a função para atualizar o gráfico com os dados obtidos.
53           atualizarGraficoRosca(dadosJson);
54           /* 'atualizarGraficoRosca' é uma função definida em outro
55           |   |   lugar no código que utiliza os dados JSON para criar ou
56           |   |   atualizar um gráfico de rosca. */
57
58       })
59       .catch(error => console.error('Erro ao carregar o arquivo Excel:', error));
60       /* 'catch' é usado para lidar com qualquer erro que ocorra
```


File Edit Selection View Go Run ... ← → P31 - Gráfico de Rosca com Dados de Excel

EXPLORER

P31 - GRÁFICO DE ROSCA COM DADOS DE EXCEL

- dados.xlsx
- # estilos.css
- index.html
- p31_grafico_rosca.png
- scripts.js

scripts.js > atualizarGraficoRosca

```
91 // Define as cores para as fatias do gráfico.  
92 var cores = [  
93     'rgba(255, 99, 132, 0.8)', // Vermelho  
94     'rgba(54, 162, 235, 0.8)', // Azul  
95     'rgba(255, 206, 86, 0.8)', // Amarelo  
96     'rgba(75, 192, 192, 0.8)', // Verde  
97     'rgba(153, 102, 255, 0.8)' // Roxo  
98 ];  
99 /* 'cores' é um array de strings representando cores em formato RGBA.  
100 | Cada cor será usada para uma fatia diferente no  
101 | | gráfico de rosca. */  
102  
103 // Define as cores das bordas para cada fatia do gráfico.  
104 var bordas = [  
105     'rgba(255, 255, 255, 1)',  
106     'rgba(255, 255, 255, 1)',  
107     'rgba(255, 255, 255, 1)',  
108     'rgba(255, 255, 255, 1)',  
109     'rgba(255, 255, 255, 1)'  
110 ];  
111 /* 'bordas' é um array com cores para as bordas das fatias do  
112 | gráfico, todas brancas neste caso,  
113 | | proporcionando um contraste limpo entre as  
114 | | fatias coloridas. */  
115  
116 // Verifica se um gráfico já existe e o destrói antes de criar um novo.  
117 if (window.meuGraficoRosca) {  
118     window.meuGraficoRosca.destroy();  
119     /* 'window.meuGraficoRosca.destroy()' remove o gráfico existente,
```

Ln 91, Col 1 Spaces: 4 UTF-8 CRLF {} JavaScript Go Live

File Edit Selection View Go Run ... ← → 🔍 P31 - Gráfico de Rosca com Dados de Excel

EXPLORER

P31 - GRÁFICO DE ROSCA COM DADOS DE EXCEL

- dados.xlsx
- # estilos.css
- index.html
- p31_grafico_rosca.png

scripts.js

index.html # estilos.css JS scripts.js X

scripts.js > atualizarGraficoRosca

```
121     se houver, para evitar sobreposições
122 quando um novo gráfico for criado. Isso é necessário
123 para garantir que as atualizações dos dados se
124 refletem corretamente. */
125
126 }
127
128 window.meuGraficoRosca = new Chart(contextoRosca, {
129
130     type: 'doughnut',
131     /* Define o tipo de gráfico como 'doughnut'. Este tipo é um
132     | gráfico circular com um centro vazio, ideal para
133     | comparar proporções. */
134
135     data: {
136         labels: produtos,
137         /* 'labels' são utilizados para identificar cada fatia do
138         | gráfico. Neste caso, representam os produtos. */
139
140         datasets: [
141             data: vendas,
142             /* 'data' contém os valores numéricos de cada produto,
143             | que serão representados nas fatias do gráfico. */
144
145             backgroundColor: cores,
146             /* 'backgroundColor' define as cores de fundo para cada
147             | fatia do gráfico, tornando-o visualmente atraente. */
148
149             borderColor: bordas,
150             /* 'borderColor' especifica a cor da borda para cada fatia do
```

Ln 121, Col 7 Spaces: 4 UTF-8 CRLF {} JavaScript ⚡ Go Live

EXPLORER

P31 - GRÁFICO DE ROSCA COM DADOS DE EXCEL

- dados.xlsx
- # estilos.css
- index.html
- p31_grafico_rosca.png
- scripts.js

...

OUTLINE

TIMELINE

index.html # estilos.css JS scripts.js X

JS scripts.js > atualizarGraficoRosca > data > datasets

```
151                                     gráfico. Aqui, todas as bordas são brancas. */
152
153             borderWidth: 2,
154             /* 'borderWidth' define a espessura da borda das fatias, em pixels. */
155
156             hoverOffset: 10
157             /* 'hoverOffset' é a distância que a fatia se desloca do
158               centro ao passar o mouse sobre ela, destacando a
159               fatia selecionada. */
160
161         }]
162     },
163
164     options: {
165         /* 'options' é o objeto onde configuramos várias opções de
166           personalização do gráfico, afetando desde a aparência
167           até o comportamento interativo do mesmo.
168           Este objeto define como os plugins e outros componentes
169           do gráfico devem operar. */
170
171     plugins: {
172         /* 'plugins' dentro de 'options' permite especificar configurações
173           para plugins individuais usados no gráfico.
174           Plugins podem adicionar funcionalidades extra ou alterar o
175           comportamento padrão do gráfico. */
176
177     title: {
178         /* 'title' é um objeto de configuração específico para o
179           plugin de título no Chart.js.
180           Este plugin gerencia a exibição de títulos em gráficos,
```


EXPLORER

P31 - GRÁFICO DE ROSCA COM DADOS DE EXCEL

- dados.xlsx
- # estilos.css
- index.html
- p31_grafico_rosca.png

scripts.js

OUTLINE

TIMELINE

index.html # estilos.css scripts.js

scripts.js > atualizarGraficoRosca > options > plugins > legend

```
211      /* A 'legend' é um componente gráfico que mostra um guia de cores ou
212      padrões associados aos dados.
213      Ela ajuda os usuários a entender o gráfico identificando
214      visualmente quais dados cada cor ou padrão representa. */
215
216      display: true,
217      /* 'display: true' especifica que a legenda deve ser exibida.
218      Quando verdadeiro, a legenda é visível ao lado do gráfico,
219      facilitando a identificação das categorias representadas
220      pelas cores das fatias do gráfico. */
221
222      position: 'right',
223      /* 'position: right' define a posição da legenda no layout do
224      gráfico. Neste caso, a legenda será posicionada à direita
225      do gráfico de rosca. Isso é útil para layouts onde há
226      espaço suficiente ao lado do gráfico e ajuda a manter o gráfico
227      e a legenda organizados de maneira limpa. */
228
229      labels: {
230          font: {
231              size: 14
232              /* 'size: 14' define o tamanho da fonte usada nos
233              textos da legenda. Um tamanho de 14 é geralmente suficiente
234              para
235              garantir boa legibilidade sem dominar o layout visual do
236              gráfico. */
237
238          },  
padding: 20
```


EXPLORER

P31 - GRÁFICO DE ROSCA COM DADOS DE EXCEL

- dados.xlsx
- # estilos.css
- index.html
- p31_grafico_rosca.png
- scripts.js

index.html # estilos.css **scripts.js** X

scripts.js > atualizarGraficoRosca > options > plugins > datalabels > font

```
354         size: 14
355         /* 'size: 14' define o tamanho da fonte dos rótulos para 14
356           pixels. Este tamanho assegura que os textos sejam
357           grandes o suficiente para serem lidos facilmente, mas
358           não tão grandes a ponto de dominar as representações gráficas.
359
360       }
361     }
362   }
363 }
364 }
365 },
366
367 plugins: [ChartDataLabels]
368 /* 'plugins' é uma opção de configuração no Chart.js que permite especificar
369    quais plugins devem ser aplicados ao gráfico.
370    - [ChartDataLabels] é uma matriz que contém as referências dos plugins
371      que serão usados. Neste caso, estamos incluindo o
372        plugin 'ChartDataLabels', que foi configurado anteriormente
373          na seção 'options'.
374 */
375
376 );
377 }
```

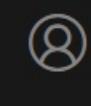
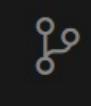
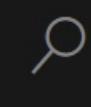
UC: Projetos

DIAS: 25/10/2024

Assunto:

Small Projects

P32 – Gráfico de Progresso de Figura



EXPLORER

▼ P32 - GRÁFICO DE PROGRESSO DE FIGURAS

base.pnc

 dados.xls

estilos.cs

index.htm

 p32 grafico progresso pp

— PostgreSQL

◀ previous

<> index.html >

<> index.html



EXPLORER

✓ P32 - GRÁFICO DE PROGRESSO DE FIGURAS

File Edit Selection View Go Run ... ← → 🔎 P32 - Gráfico de Progresso de Figura

EXPLORER ...

P32 - GRÁFICO DE PROGRESCO DE FIGURA

- base.png
- dados.xlsx
- # estilos.css
- index.html
- p32_grafico_progresso.png
- preenchimento.png
- scripts.js

index.html

index.html > html > body > div.container

```
61      <h1>Gráfico de Progresso de Figura</h1>
62      <!-- A tag &lt;h1&gt; é usada aqui como o título principal da página.
63          Ela fornece uma visão clara do conteúdo ou propósito
64          principal da página, neste caso, um gráfico de
65          progresso de figura. --&gt;
66
67      &lt;select id="seletorProduto"&gt;
68          <!-- A tag &lt;select&gt; cria uma caixa de seleção dropdown.
69              O atributo 'id' facilita a manipulação deste
70              elemento via JavaScript. --&gt;
71          &lt;!-- Opções de produtos serão carregadas aqui --&gt;
72          &lt;!-- Este comentário indica que as opções dentro do &lt;select&gt;
73              serão preenchidas dinamicamente com JavaScript,
74              com base nos dados carregados de uma fonte externa ou arquivo. --&gt;
75
76      &lt;/select&gt;
77
78      &lt;div id="descricaoProduto"&gt;&lt;/div&gt;
79      <!-- Uma &lt;div&gt; vazia com o 'id' de 'descricaoProduto'. Esta div
80          será usada para exibir informações dinâmicas sobre o
81          produto selecionado, tais como detalhes adicionais
82          ou uma descrição. --&gt;
83
84      &lt;div class="grafico-container"&gt;
85          <!-- A classe 'grafico-container' indica que esta &lt;div&gt; é
86              usada para conter elementos relacionados ao
87              gráfico de progresso. --&gt;
88
89          &lt;img id="imagemBase" src="base.png" alt="Imagem Base"&gt;</pre>

Ln 61, Col 1 Spaces: 4 UTF-8 CRLF HTML Go Live


```


The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P32 - Gráfico de Progresso de Figura
- Icons:** Explorer, Find, Open, Save, Help, Outline, Timeline.
- Left Sidebar (EXPLORER):** P32 - GRÁFICO DE PROGRESCO DE FIGURA (expanded), containing files: base.png, dados.xlsx, # estilos.css (selected), index.html, p32_grafico_progresso.png, preenchimento.png, scripts.js.
- Central Area:** Editor showing index.html and # estilos.css tabs. The # estilos.css tab is active, displaying the following CSS code:

```
body {  
    font-family: 'Roboto', sans-serif;  
    /* Define a família de fontes para 'Roboto', seguida  
       de 'sans-serif' como fallback. 'Roboto' é uma fonte  
       moderna e legível, adequada para interfaces digitais. */  
  
    background-color: #f4f4f4;  
    /* Define a cor de fundo do corpo do documento como um cinza  
       claro (#f4f4f4), proporcionando um fundo neutro que não  
       distrai, ideal para conteúdos coloridos ou interfaces. */  
  
    color: #333;  
    /* Define a cor padrão do texto para um cinza escuro (#333), que  
       oferece bom contraste com o fundo claro, melhorando a legibilidade. */  
  
    display: flex;  
    /* Ativa o modelo de layout Flexbox, que permite um alinhamento  
       mais fácil e responsivo dos elementos filhos. */  
  
    flex-direction: column;  
    /* Define a direção principal do layout flexível para 'column',  
       fazendo com que os elementos filhos sejam organizados verticalmente. */  
  
    align-items: center;  
    /* Alinha os elementos filhos ao centro do eixo transversal  
       (horizontalmente neste caso), centralizando-os dentro do corpo. */  
  
    margin: 0;  
    /* Remove todas as margens externas do elemento 'body',
```

Bottom status bar: ⌂ 0 △ 0 ⌂ 0 Ln 1, Col 1 Spaces: 8 UTF-8 CRLF CSS ⌂ Go Live ⌂

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P32 - Gráfico de Progresso de Figura
- Icons:** Explorer, Find, Open, Save, Help, Outline, Timeline.
- Left Sidebar (EXPLORER):** P32 - GRÁFICO DE PROGRESCO DE FIGURA (selected), base.png, dados.xlsx, # estilos.css, index.html, p32_grafico_progresso.png, preenchimento.png, scripts.js.
- Active Editor:** # estilos.css (highlighted)
- Code Content:** CSS rules for styling a container element.

```
index.html # estilos.css

# estilos.css > body
31     permitindo que o conteúdo utilize todo o espaço disponível
32         na janela do navegador. */
33
34     padding: 20px;
35     /* Adiciona um preenchimento de 20px ao redor do conteúdo interno
36         do corpo, criando um espaço entre o conteúdo e as bordas do
37         navegador, aumentando a área de respiração visual. */
38
39 }
40
41 .container {
42
43     text-align: center;
44     /* Alinha o texto e, por extensão, outros elementos inline ao
45         centro do contêiner. Isso é útil para garantir que títulos,
46         textos e outros conteúdos sejam centralizados visualmente. */
47
48     background-color: #fff;
49     /* Define a cor de fundo do elemento '.container' como branco (#fff),
50         criando um contraste claro com o fundo cinza do 'body' e
51         destacando visualmente a área principal de conteúdo. */
52
53     padding: 20px;
54     /* Adiciona um preenchimento de 20px ao redor do conteúdo dentro
55         do '.container', aumentando o espaço entre o conteúdo e as
56         bordas do contêiner, melhorando a legibilidade e a estética. */
57
58     border-radius: 10px;
59     /* Arredonda os cantos do contêiner com um raio de 10px, proporcionando
60         uma aparência mais suave e moderna, que é visualmente
```

A screenshot of a code editor interface, likely Microsoft Visual Studio Code, displaying a CSS file for a progress bar figure. The top navigation bar includes File, Edit, Selection, View, Go, Run, and a search bar labeled 'P32 - Gráfico de Progresso de Figura'. The left sidebar features icons for Explorer, Search, Find, Preview, and Outline/Timeline. The Explorer panel shows files like base.png, dados.xlsx, #estilos.css, index.html, p32_grafico_progresso.png, preenchimento.png, and scripts.js. The main editor area shows the following CSS code:

```
index.html # estilos.css
# estilos.css > .container
61     agradável e menos formal. */
62
63     box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
64     /* Aplica uma sombra suítil ao redor do '.container'. Os valores
65      especificam o deslocamento horizontal, vertical, o raio de
66      desfoque, e a cor da sombra (um preto com 10% de transparência),
67      criando um efeito de elevação visual que faz o contêiner
68      parecer flutuar acima do restante da página. */
69
70     max-width: 800px;
71     /* Define a largura máxima do '.container' para 800px. Isso
72      garante que o contêiner não se torne excessivamente largo
73      em telas grandes, mantendo a legibilidade e a estética do layout. */
74
75     width: 100%;
76     /* Define a largura do '.container' para ocupar 100% do espaço
77      disponível de seu elemento pai, até o máximo de 800px. Isso
78      garante que o contêiner se adapte ao tamanho da tela ou ao
79      espaço disponível. */
80
81 }
82
83 .grafico-container {
84
85     position: relative;
86     /* Define a posição do container como relativa, o que permite
87      que os elementos filhos com posição absoluta se
88      posicionem em relação a ele, criando um contexto de
89      posicionamento localizado para seus elementos internos. */
90 }
```

File Edit Selection View Go Run ... ← → 🔍 P32 - Gráfico de Progresso de Figura

EXPLORER ...

P32 - GRÁFICO DE PROGRESCO DE FIGURA

- base.png
- dados.xlsx
- # estilos.css
- index.html
- p32_grafico_progresso.png
- preenchimento.png
- scripts.js

estilos.css # estilos.css .grafico-container

```
91     width: 400px;
92     /* Estabelece a largura do container do gráfico para 400 pixels,
93      garantindo que o gráfico tenha um tamanho fixo e adequado
94      para a visualização das imagens de progresso. */
95
96     height: 400px;
97     /* Define a altura do container do gráfico, também para 400 pixels,
98      criando uma área quadrada perfeita que facilita o
99      alinhamento e a proporção visual das imagens internas. */
100
101    margin: 0 auto;
102    /* Centraliza o container do gráfico horizontalmente dentro de
103       seu elemento pai, utilizando '0' para a margem superior e
104       inferior e 'auto' para as margens laterais, otimizando a
105       visualização central no layout da página. */
106
107 }
108
109
110 .grafico-container img {
111
112     position: absolute;
113     /* A posição absoluta é aplicada para cada imagem dentro do
114        container, fazendo com que elas se posicionem em relação
115        ao próprio container, devido à sua posição relativa. Isso é
116        essencial para sobrepor as imagens de base e
117        preenchimento corretamente. */
118
119     top: 0;
120     /* Posiciona as imagens no topo do container, garantindo que o
```

Ln 91, Col 4 Spaces: 8 UTF-8 CRLF CSS ⚡ Go Live

File Edit Selection View Go Run ... ⏪ ⏩ 🔎 P32 - Gráfico de Progresso de Figura

EXPLORER ...

P32 - GRÁFICO DE PROGRESCO DE FIGURA

- base.png
- dados.xlsx
- # estilos.css
- index.html
- p32_grafico_progresso.png
- preenchimento.png
- scripts.js

estilos.css # estilos.css .grafico-container img

```
121     ponto de início vertical seja o canto superior do container. */
122
123     left: 0;
124     /* Posiciona as imagens à esquerda do container, garantindo que o
125        ponto de início horizontal seja o canto esquerdo do container. */
126
127     width: 100%;
128     /* Define a largura das imagens para cobrir 100% da largura do
129        container, assegurando que as imagens se estendam por
130        toda a largura disponível, mantendo a proporção e o
131        alinhamento corretos. */
132
133     height: 100%;
134     /* Define a altura das imagens para cobrir 100% da altura do
135        container, garantindo que as imagens preencham
136        completamente a altura disponível, alinhando-se
137        perfeitamente ao espaço designado. */
138
139 }
140
141 #percentual {
142
143     position: absolute;
144     /* A propriedade 'position: absolute' é usada para posicionar o
145        elemento de forma absoluta em relação ao seu contêiner
146        posicionado mais próximo, neste caso, dentro do ` `.grafico-container` ,
147        que tem 'position: relative'. Isso permite que o percentual seja
148        posicionado exatamente sobre a imagem do gráfico. */
149
150     top: 50%;
```

Ln 121, Col 1 Spaces: 8 UTF-8 CRLF CSS 🔍 Go Live

The screenshot shows a code editor interface with the following details:

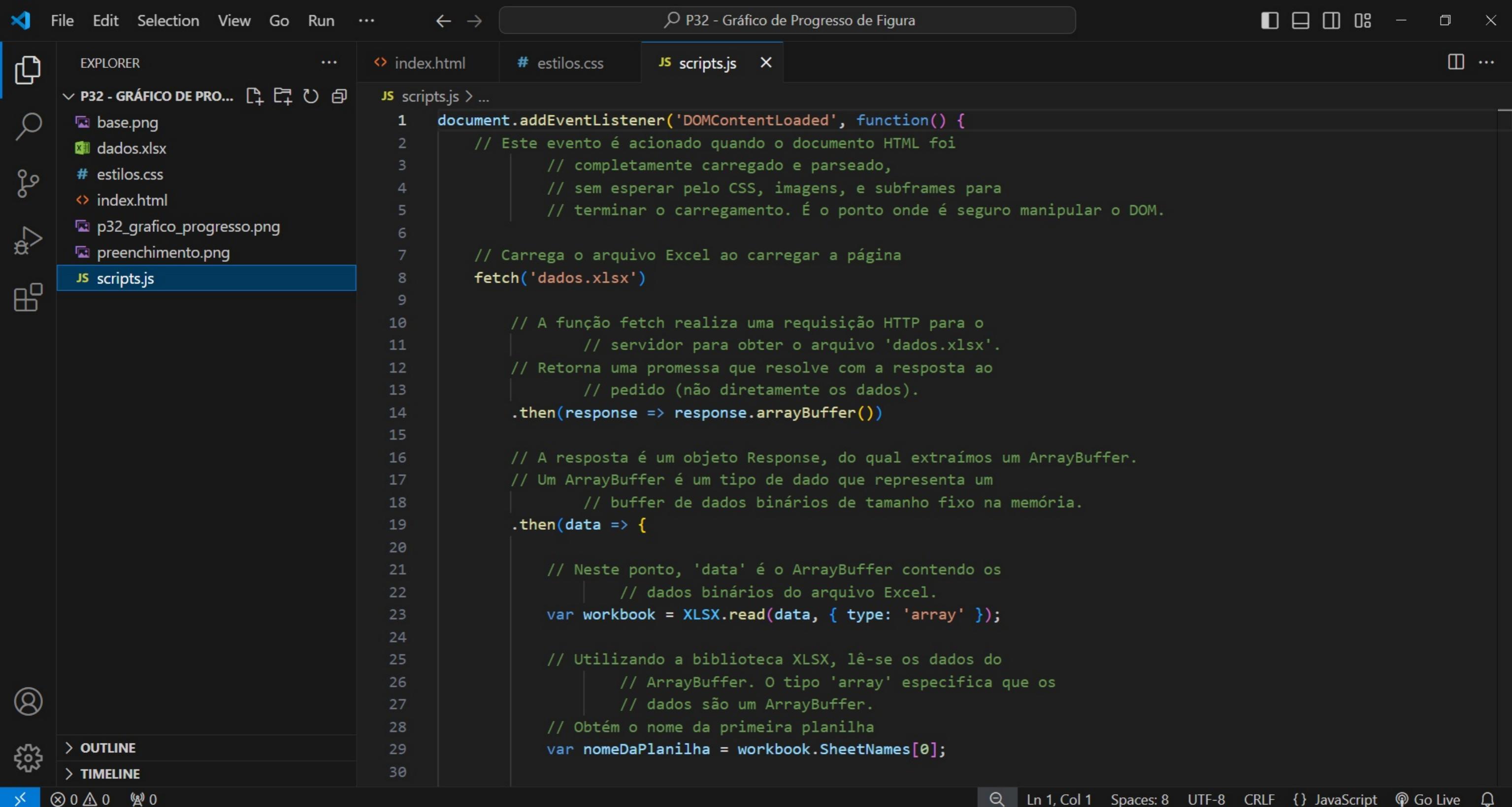
- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P32 - Gráfico de Progresso de Figura
- Icons:** Explorer, Search, Find, Open, Save, Outline, Timeline.
- Left Sidebar:** EXPLORER, P32 - GRÁFICO DE PROGRESCO DE FIGURA (base.png, dados.xlsx, #estilos.css, index.html, p32_grafico_progresso.png, preenchimento.png, scripts.js).
- Active Editor:** # estilos.css (highlighted in blue)
- Code Content:** CSS code for styling a progress bar figure. The code includes:
 - Positioning: left: 50%; transform: translate(-50%, -50%);
 - Font Size: font-size: 24px;
 - Color: color: #333;
- Bottom Status Bar:** x 0 △ 0 ⌂ 0, Ln 151, Col 4, Spaces: 8, UTF-8, CRLF, CSS, Go Live, ⌂

```
< > index.html # estilos.css X
# estilos.css > #percentual
151    left: 50%;
152    /* As propriedades 'top' e 'left' definem a posição do elemento a 50%
153       do topo e 50% da esquerda do seu contêiner, posicionando-o
154       teoricamente no centro. */
155
156    transform: translate(-50%, -50%);
157    /* 'transform: translate(-50%, -50%)' ajusta o posicionamento,
158       movendo o elemento para trás pela metade de sua própria
159       altura e largura, garantindo que ele seja centralizado
160       precisamente no meio do gráfico. Este método de centralização é
161       muito eficaz e responsivo. */
162
163    font-size: 24px;
164    /* Define o tamanho da fonte para 24 pixels, garantindo que o
165       texto seja grande o suficiente para ser facilmente
166       legível sobre o gráfico de progresso. */
167
168    color: #333;
169    /* Define a cor do texto para um cinza escuro (#333), que oferece
170       excelente contraste com a maioria dos fundos, melhorando a
171       legibilidade do texto sobre imagens coloridas ou
172       variadas no gráfico. */
173
174 }
175
176 #descricaoProduto {
177
178     margin-top: 20px;
179     /* Adiciona uma margem superior de 20px, separando visualmente a
180       descrição do produto dos elementos acima dela, como o
```

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P32 - Gráfico de Progresso de Figura
- Explorer:** Shows files and folders:
 - P32 - GRÁFICO DE PROGRESCO DE FIGURA
 - base.png
 - dados.xlsx
 - # estilos.css
 - index.html
 - p32_grafico_progresso.png
 - preenchimento.png
 - scripts.js
- Code Editor:** The # estilos.css file is open, showing the following CSS code:

```
181 # descricaoProduto {  
182     /* Define o tamanho da fonte para 18 pixels, que é suficientemente  
183     grande para facilitar a leitura sem dominar visualmente  
184     outros elementos da interface. */  
185     font-size: 18px;  
186     /* Configura a cor do texto para um cinza escuro (#333). Esta  
187     cor foi escolhida para manter a consistência com outras  
188     partes do texto na página e para assegurar boa  
189     visibilidade e contraste. */  
190     color: #333;  
191 }  
192 }  
193 }  
194 }  
195 }
```
- Bottom Status Bar:** Shows file count (0), line count (Ln 181, Col 8), spaces (Spaces: 8), encoding (UTF-8), line endings (CRLF), CSS, Go Live, and a bell icon.



The screenshot shows a Microsoft Edge browser window with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Address Bar:** P32 - Gráfico de Progresso de Figura
- Icons:** Back, Forward, Stop, Refresh, Home, Favorites, Help.
- Left Sidebar:** EXPLORER, P32 - GRÁFICO DE PRO..., base.png, dados.xlsx, # estilos.css, index.html, p32_grafico_progresso.png, preenchimento.png, JS scripts.js (selected).
- Content Area:** A code editor showing JavaScript code for reading an Excel file. The code uses the XLSX library to read an ArrayBuffer from a fetched XLSX file and extract the first sheet name.

```
document.addEventListener('DOMContentLoaded', function() {
    // Este evento é acionado quando o documento HTML foi
    // completamente carregado e parseado,
    // sem esperar pelo CSS, imagens, e subframes para
    // terminar o carregamento. É o ponto onde é seguro manipular o DOM.

    // Carrega o arquivo Excel ao carregar a página
    fetch('dados.xlsx')

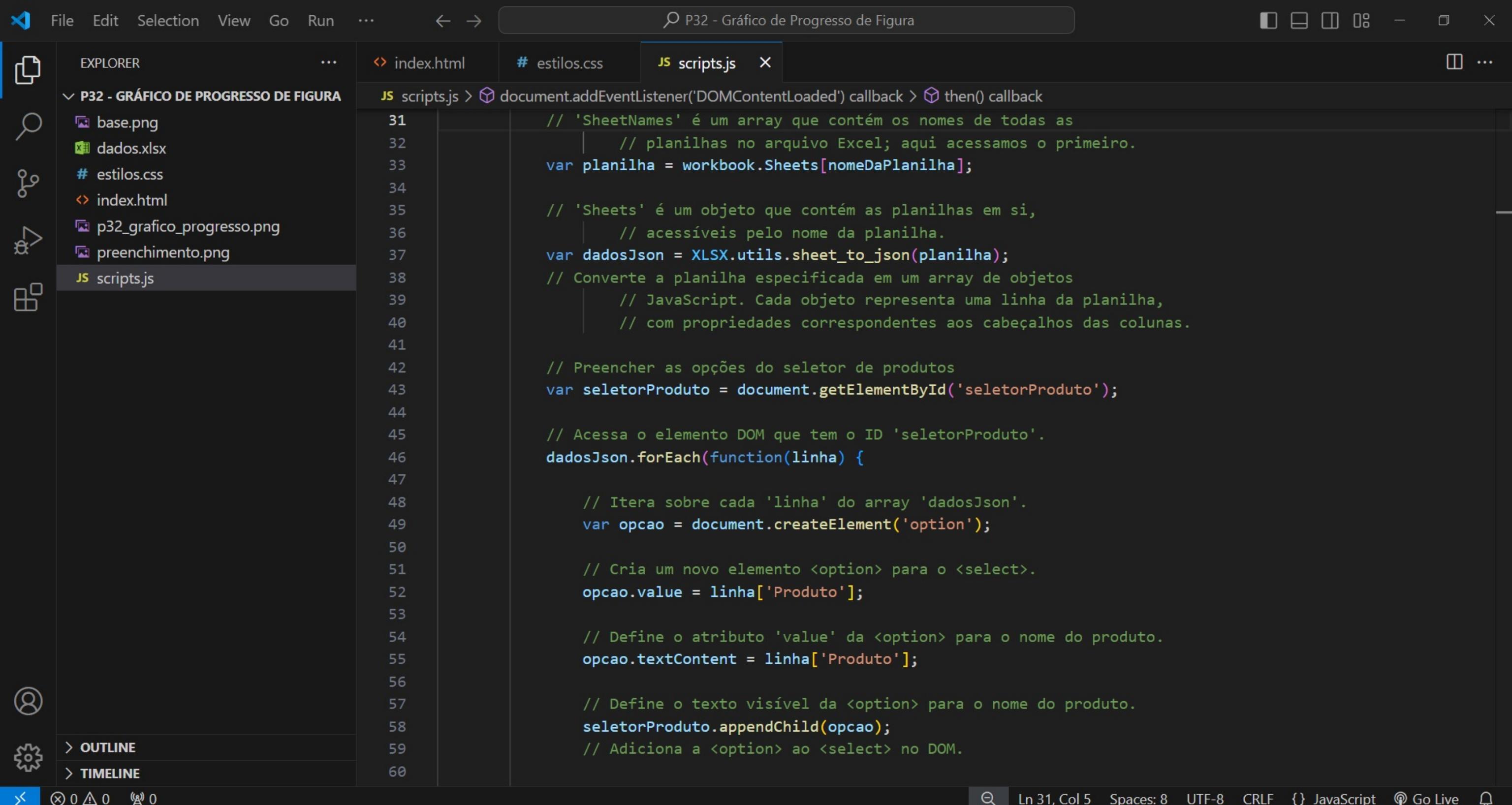
    // A função fetch realiza uma requisição HTTP para o
    // servidor para obter o arquivo 'dados.xlsx'.
    // Retorna uma promessa que resolve com a resposta ao
    // pedido (não diretamente os dados).
    .then(response => response.arrayBuffer())

    // A resposta é um objeto Response, do qual extraímos um ArrayBuffer.
    // Um ArrayBuffer é um tipo de dado que representa um
    // buffer de dados binários de tamanho fixo na memória.
    .then(data => {

        // Neste ponto, 'data' é o ArrayBuffer contendo os
        // dados binários do arquivo Excel.
        var workbook = XLSX.read(data, { type: 'array' });

        // Utilizando a biblioteca XLSX, lê-se os dados do
        // ArrayBuffer. O tipo 'array' especifica que os
        // dados são um ArrayBuffer.
        // Obtém o nome da primeira planilha
        var nomeDaPlanilha = workbook.SheetNames[0];
    })
})
```

- Bottom Bar:** Navigation icons (Home, Back, Forward, Stop), Address bar, Ln 1, Col 1, Spaces: 8, UTF-8, CRLF, {} JavaScript, Go Live, Timeline.



The screenshot shows a Microsoft Edge browser window with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Address Bar:** P32 - Gráfico de Progresso de Figura
- Icons:** Back, Forward, Stop, Refresh, Home, Favorites, Help.
- Left Sidebar:** EXPLORER, P32 - GRÁFICO DE PROGRESCO DE FIGURA (containing base.png, dados.xlsx, # estilos.css, index.html, p32_grafico_progresso.png, preenchimento.png, scripts.js), OUTLINE, and TIMELINE.
- Current Tab:** scripts.js
- Code Editor Content:**

```
// 'SheetNames' é um array que contém os nomes de todas as
// planilhas no arquivo Excel; aqui acessamos o primeiro.
var planilha = workbook.Sheets[nomeDaPlanilha];

// 'Sheets' é um objeto que contém as planilhas em si,
// acessíveis pelo nome da planilha.
var dadosJson = XLSX.utils.sheet_to_json(planilha);
// Converte a planilha especificada em um array de objetos
// JavaScript. Cada objeto representa uma linha da planilha,
// com propriedades correspondentes aos cabeçalhos das colunas.

// Preencher as opções do seletor de produtos
var seletorProduto = document.getElementById('seletorProduto');

// Acessa o elemento DOM que tem o ID 'seletorProduto'.
dadosJson.forEach(function(linha) {

    // Itera sobre cada 'linha' do array 'dadosJson'.
    var opcao = document.createElement('option');

    // Cria um novo elemento <option> para o <select>.
    opcao.value = linha['Produto'];

    // Define o atributo 'value' da <option> para o nome do produto.
    opcao.textContent = linha['Produto'];

    // Define o texto visível da <option> para o nome do produto.
    seletorProduto.appendChild(opcao);
    // Adiciona a <option> ao <select> no DOM.
```
- Bottom Bar:** Icons for search, refresh, and other browser functions, followed by Ln 31, Col 5, Spaces: 8, UTF-8, CRLF, {}, JavaScript, Go Live, and a lock icon.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top bar includes standard menu items: File, Edit, Selection, View, Go, Run, and three ellipsis buttons. To the right of the menu is a search bar containing the text "P32 - Gráfico de Progresso de Figura". On the far right are window control buttons (minimize, maximize, close) and a set of icons for document preview, file operations, and other settings.

The left sidebar features the "EXPLORER" view, which lists project files: "base.png", "dados.xlsx", "# estilos.css", "index.html", "p32_grafico_progresso.png", "preenchimento.png", and "scripts.js". The "scripts.js" file is currently selected, indicated by a dark grey background.

The main editor area displays the "scripts.js" code. The code uses the DOMContentLoaded event to run a function that updates a chart based on a selected product. It adds a change event listener to a select element, finds the selected product in a JSON array, and then calls a function to update the chart with the found product's data.

```
File Edit Selection View Go Run ...
← → ⌂ P32 - Gráfico de Progresso de Figura
EXPLORE ...
P32 - GRÁFICO DE PROGRESCO DE FIGURA
base.png
dados.xlsx
# estilos.css
index.html
p32_grafico_progresso.png
preenchimento.png
scripts.js

scripts.js > document.addEventListener('DOMContentLoaded') callback > then() callback > dadosJson.forEach() callback

61   });
62
63     // Atualizar gráfico com base na seleção do produto
64     seletorProduto.addEventListener('change', function() {
65
66       // Adiciona um ouvinte de evento 'change' ao elemento select 'seletorProduto'.
67       // Este evento é disparado cada vez que o usuário altera a seleção no dropdown.
68       var produtoSelecionado = this.value;
69       // 'this.value' refere-se ao valor do produto atualmente
70         // selecionado no <select>, que é o valor do
71         // atributo 'value' da <option> selecionada.
72
73       var dadosProduto = dadosJson.find(item => item['Produto'] === produtoSelecionado);
74       // Utiliza o método 'find' do array para procurar no
75         // array 'dadosJson' o primeiro elemento
76         // onde a propriedade 'Produto' é igual ao produto
77         // selecionado. Retorna o objeto completo
78         // que representa o produto e seus dados associados.
79
80       if (dadosProduto) {
81
82         atualizarGrafico(dadosProduto);
83         // Se um produto correspondente é encontrado, chama a
84           // função 'atualizarGrafico', passando o objeto
85             // com os dados do produto selecionado. Essa função é responsável
86               // por atualizar o gráfico visual na página com
87                 // base nos dados do produto.
88
89     }
90   });

Ln 61, Col 1 Spaces: 8 UTF-8 CRLF {} JavaScript ⌂ Go Live
```

File Edit Selection View Go Run ... ← → P32 - Gráfico de Progresso de Figura

EXPLORER ...

P32 - GRÁFICO DE PROGRESCO DE FIGURA

- base.png
- dados.xlsx
- # estilos.css
- index.html
- p32_grafico_progresso.png
- preenchimento.png
- scripts.js

scripts.js > document.addEventListener('DOMContentLoaded') callback > then() callback

```
91 // Inicializar gráfico com o primeiro produto
92 if (dadosJson.length > 0) {
93     // Verifica se o array 'dadosJson' contém algum
94     // elemento, garantindo que existem dados para serem processados.
95     seletorProduto.value = dadosJson[0]['Produto'];
96     // Define o valor do <select> para o nome do primeiro
97     // produto no array 'dadosJson'.
98     // Isso configura o dropdown para mostrar o primeiro
99     // produto como selecionado ao carregar a página.
100
101    atualizarGrafico(dadosJson[0]);
102    // Chama a função 'atualizarGrafico' para o primeiro
103    // produto no array, garantindo que
104    // o gráfico seja inicializado com dados
105    // assim que a página é carregada.
106
107
108
109 }
110 }
111 .catch(error => console.error('Erro ao carregar o arquivo Excel:', error));
112 // O método 'catch' é usado para capturar qualquer erro
113 // que ocorra durante o processo de fetch
114 // ou processamento de dados. Se um erro ocorrer,
115 // ele será logado no console do navegador,
116 // ajudando no diagnóstico e resolução de problemas.
117
118 });
119
120 function atualizarGrafico(dadosProduto) {
```

OUTLINE

TIMELINE

x 0 △ 0 ⌂ 0 🔍 Ln 91, Col 1 Spaces: 8 UTF-8 CRLF {} JavaScript Go Live

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P32 - Gráfico de Progresso de Figura
- Explorer:** Shows a project structure under P32 - GRÁFICO DE PROGRESCO DE FIGURA, including files like base.png, dados.xlsx, # estilos.css, index.html, p32_grafico_progresso.png, preenchimento.png, and scripts.js.
- Code Editor:** The scripts.js file is open, displaying the following JavaScript code:

```
121 var meta = 100; // Meta fixa de 100
122 // Define uma meta de vendas constante como 100 para
123 // simplificar o cálculo do percentual de preenchimento do gráfico.
124
125 var vendas = dadosProduto['Vendas'];
126 // Extrai a quantidade de vendas do produto do objeto 'dadosProduto',
127 // que contém dados específicos do produto selecionado.
128
129 var percentual = (vendas / meta) * 100;
130 // Calcula o percentual de vendas em relação à meta. Isso
131 // determina até que ponto a imagem de preenchimento
132 // deverá ser preenchida.
133
134 var imagemPreenchimento = document.getElementById('imagemPreenchimento');
135 // Obtém a referência à imagem de preenchimento no DOM, que será
136 // manipulada para refletir visualmente o progresso das vendas.
137
138 var percentualDiv = document.getElementById('percentual');
139 // Acessa o elemento DOM que exibe o percentual numericamente,
140 // permitindo atualizar seu conteúdo.
141
142 var descricaoProduto = document.getElementById('descricaoProduto');
143 // Acessa o elemento DOM que exibe a descrição do produto, para
144 // atualizar os detalhes do produto conforme a seleção.
145
146 // Ajustar a altura da imagem de preenchimento com base no percentual
147 imagemPreenchimento.style.clipPath = `inset(${100 - percentual}% 0 0 0)`;
148 // Aplica um estilo 'clipPath' à imagem de preenchimento para
149 // criar um efeito visual que mostra apenas uma parte da imagem
150
```

The code implements a progress bar visualization by calculating the percentage of sales relative to a fixed target (100). It then uses CSS's `clip-path` property to show only the portion of an image corresponding to that percentage.

The screenshot shows a code editor interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** P32 - Gráfico de Progresso de Figura
- Icons:** Explorer, Search, Open, Close, Help, Outline, Timeline.
- Left Sidebar (EXPLORER):** P32 - GRÁFICO DE PROGRESCO DE FIGURA (expanded), containing files: base.png, dados.xlsx, #estilos.css, index.html, p32_grafico_progresso.png, preenchimento.png, scripts.js.
- Active Tab:** scripts.js
- Code Content:** A snippet of JavaScript code for updating a progress bar chart.

```
// baseada no percentual de vendas. O 'inset' é calculado como 100
// menos o percentual calculado, efetivamente revelando a
// parte da imagem que corresponde ao desempenho das vendas.

// Atualizar texto percentual
percentualDiv.textContent = `${percentual.toFixed(2)}%`;
// Atualiza o texto dentro do elemento 'percentualDiv' para
// mostrar o percentual de vendas com duas casas decimais,
// seguido de um símbolo de porcentagem.

// Atualizar descrição do produto
descricaoProduto.textContent = `Produto: ${dadosProduto['Produto']} - Total de Vendas: ${vendas}`;
// Atualiza o texto na 'descricaoProduto' para incluir o nome do
// produto e o total de vendas, fornecendo contexto adicional
// sobre o produto selecionado.

}
```
- Bottom Status Bar:** Line 151, Col 6, Spaces: 8, UTF-8, CRLF, {}, JavaScript, Go Live.