

Atividade para dia 06/09

Centro Universitário - UNIESP

Ciência da Computação

Tecnologia para Fron-End

Leandro Melo

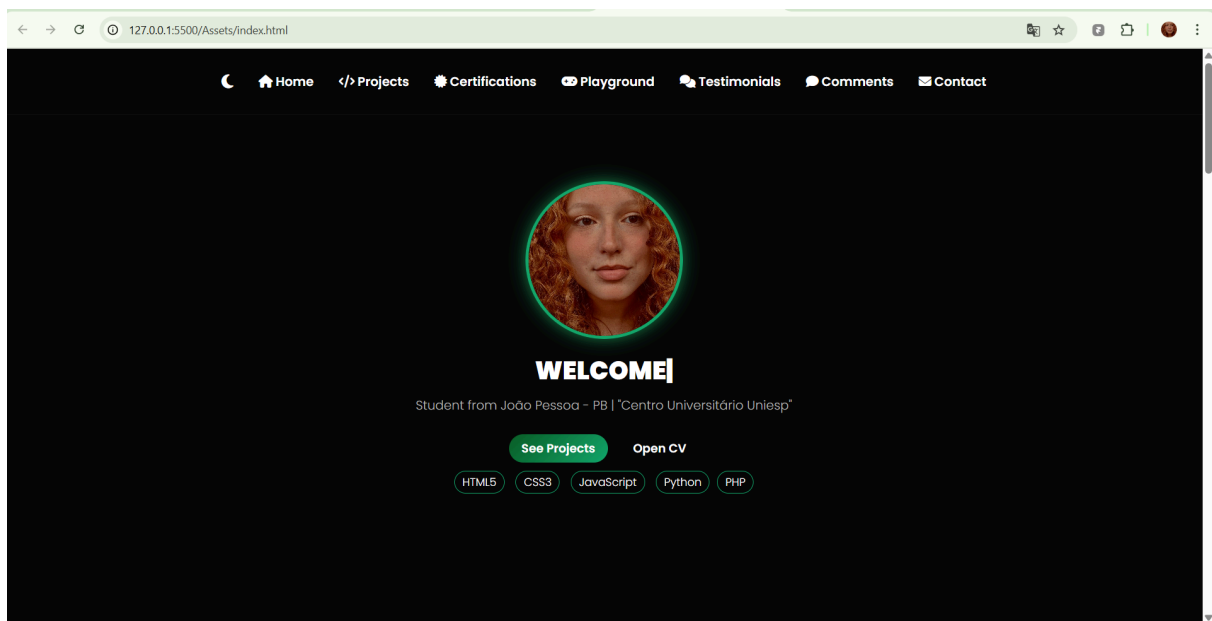
Helena Trombetta - 2025211880176

Questão 1. Explique com suas palavras o que é HTML e qual o seu papel no desenvolvimento de páginas web.

HTML (hypertext markup language) é uma linguagem de marcação, isto é, utiliza de tags para promover uma web aplicação.

Ele é a base de toda a estrutura de um site.

Minha recente aplicação Web:



Questão 2. Quem foi o criador do HTML e em que ano a linguagem foi formalizada em especificações?

Tim Berners-Lee criou o HTML e em 1990 foi feita em especificações formais.

```
HTML: É a estrutura.
CSS: É a parte do estilo do
site.
JS: É a parte de animações do
site.
```

[illegible]

```

1  // script.js
2
3  // Elements
4
5  // First, I grab references to the main DOM elements I will
6  // use. I'm using the document.getElementById() technique
7  // here, but you could use document.querySelector() or
8  // document.getElementsByClassName() as well.
9
10 // document.getElementById() returns a reference to the document contained in the window.
11
12 const body = document.getElementById('body');
13 const lightToggle = document.getElementById('lightToggle');
14 const playMusic = document.getElementById('playground-music');
15 const stopMusic = document.getElementById('stop-music');
16 const themePicker = document.getElementById('theme-picker');
17 const form = document.getElementById('contact-form');
18
19 // DOMContentLoaded Event
20
21 // DOM Content Loaded
22
23 // I want until the page is fully loaded before running
24 // my main logic, ensuring all elements are available.
25
26 document.addEventListener('DOMContentLoaded', () => {
27
28   // Main Function
29
30   // Case Study
31
32   // I check localstorage for a saved theme preference.
33   // If "light", I apply light theme; otherwise, dark theme.
34   // The better logic flows, updating both the class
35   // on the body and the icon (element).
36   // Your preference is stored in localstorage.
37
38   const themeName = localStorage.getItem('theme');
39
40   if (themeName === 'light') {
41     // Now I check for any saved light
42
43     // I check localstorage for a saved music preference.
44     // If "on", I apply music; otherwise, I don't.
45     // The better logic flows, updating both the class
46     // on the body and the icon (element).
47     // Your preference is stored in localstorage.
48
49     const musicName = localStorage.getItem('music');
50
51     if (musicName === 'on') {
52       // Now I check for any saved music
53
54       // I check localstorage for a saved contact form preference.
55       // If "on", I apply the form; otherwise, I don't.
56       // The better logic flows, updating both the class
57       // on the body and the icon (element).
58       // Your preference is stored in localstorage.
59
60       const contactName = localStorage.getItem('contact');
61
62       if (contactName === 'on') {
63         // Now I check for any saved contact
64
65       }
66     }
67   }
68
69   // I check localstorage for a saved contact form preference.
70   // If "on", I apply the form; otherwise, I don't.
71   // The better logic flows, updating both the class
72   // on the body and the icon (element).
73   // Your preference is stored in localstorage.
74
75   const contactName = localStorage.getItem('contact');
76
77   if (contactName === 'on') {
78     // Now I check for any saved contact
79
80   }
81
82   // I check localstorage for a saved contact form preference.
83   // If "on", I apply the form; otherwise, I don't.
84   // The better logic flows, updating both the class
85   // on the body and the icon (element).
86   // Your preference is stored in localstorage.
87
88   const contactName = localStorage.getItem('contact');
89
90   if (contactName === 'on') {
91     // Now I check for any saved contact
92
93   }
94
95   // I check localstorage for a saved contact form preference.
96   // If "on", I apply the form; otherwise, I don't.
97   // The better logic flows, updating both the class
98   // on the body and the icon (element).
99   // Your preference is stored in localstorage.
100
101   const contactName = localStorage.getItem('contact');
102
103   if (contactName === 'on') {
104     // Now I check for any saved contact
105
106   }
107
108   // I check localstorage for a saved contact form preference.
109   // If "on", I apply the form; otherwise, I don't.
110   // The better logic flows, updating both the class
111   // on the body and the icon (element).
112   // Your preference is stored in localstorage.
113
114   const contactName = localStorage.getItem('contact');
115
116   if (contactName === 'on') {
117     // Now I check for any saved contact
118
119   }
120
121   // I check localstorage for a saved contact form preference.
122   // If "on", I apply the form; otherwise, I don't.
123   // The better logic flows, updating both the class
124   // on the body and the icon (element).
125   // Your preference is stored in localstorage.
126
127   const contactName = localStorage.getItem('contact');
128
129   if (contactName === 'on') {
130     // Now I check for any saved contact
131
132   }
133
134   // I check localstorage for a saved contact form preference.
135   // If "on", I apply the form; otherwise, I don't.
136   // The better logic flows, updating both the class
137   // on the body and the icon (element).
138   // Your preference is stored in localstorage.
139
140   const contactName = localStorage.getItem('contact');
141
142   if (contactName === 'on') {
143     // Now I check for any saved contact
144
145   }
146
147   // I check localstorage for a saved contact form preference.
148   // If "on", I apply the form; otherwise, I don't.
149   // The better logic flows, updating both the class
150   // on the body and the icon (element).
151   // Your preference is stored in localstorage.
152
153   const contactName = localStorage.getItem('contact');
154
155   if (contactName === 'on') {
156     // Now I check for any saved contact
157
158   }
159
160   // I check localstorage for a saved contact form preference.
161   // If "on", I apply the form; otherwise, I don't.
162   // The better logic flows, updating both the class
163   // on the body and the icon (element).
164   // Your preference is stored in localstorage.
165
166   const contactName = localStorage.getItem('contact');
167
168   if (contactName === 'on') {
169     // Now I check for any saved contact
170
171   }
172
173   // I check localstorage for a saved contact form preference.
174   // If "on", I apply the form; otherwise, I don't.
175   // The better logic flows, updating both the class
176   // on the body and the icon (element).
177   // Your preference is stored in localstorage.
178
179   const contactName = localStorage.getItem('contact');
180
181   if (contactName === 'on') {
182     // Now I check for any saved contact
183
184   }
185
186   // I check localstorage for a saved contact form preference.
187   // If "on", I apply the form; otherwise, I don't.
188   // The better logic flows, updating both the class
189   // on the body and the icon (element).
190   // Your preference is stored in localstorage.
191
192   const contactName = localStorage.getItem('contact');
193
194   if (contactName === 'on') {
195     // Now I check for any saved contact
196
197   }
198
199   // I check localstorage for a saved contact form preference.
200   // If "on", I apply the form; otherwise, I don't.
201   // The better logic flows, updating both the class
202   // on the body and the icon (element).
203   // Your preference is stored in localstorage.
204
205   const contactName = localStorage.getItem('contact');
206
207   if (contactName === 'on') {
208     // Now I check for any saved contact
209
210   }
211
212   // I check localstorage for a saved contact form preference.
213   // If "on", I apply the form; otherwise, I don't.
214   // The better logic flows, updating both the class
215   // on the body and the icon (element).
216   // Your preference is stored in localstorage.
217
218   const contactName = localStorage.getItem('contact');
219
220   if (contactName === 'on') {
221     // Now I check for any saved contact
222
223   }
224
225   // I check localstorage for a saved contact form preference.
226   // If "on", I apply the form; otherwise, I don't.
227   // The better logic flows, updating both the class
228   // on the body and the icon (element).
229   // Your preference is stored in localstorage.
230
231   const contactName = localStorage.getItem('contact');
232
233   if (contactName === 'on') {
234     // Now I check for any saved contact
235
236   }
237
238   // I check localstorage for a saved contact form preference.
239   // If "on", I apply the form; otherwise, I don't.
240   // The better logic flows, updating both the class
241   // on the body and the icon (element).
242   // Your preference is stored in localstorage.
243
244   const contactName = localStorage.getItem('contact');
245
246   if (contactName === 'on') {
247     // Now I check for any saved contact
248
249   }
250
251   // I check localstorage for a saved contact form preference.
252   // If "on", I apply the form; otherwise, I don't.
253   // The better logic flows, updating both the class
254   // on the body and the icon (element).
255   // Your preference is stored in localstorage.
256
257   const contactName = localStorage.getItem('contact');
258
259   if (contactName === 'on') {
260     // Now I check for any saved contact
261
262   }
263
264   // I check localstorage for a saved contact form preference.
265   // If "on", I apply the form; otherwise, I don't.
266   // The better logic flows, updating both the class
267   // on the body and the icon (element).
268   // Your preference is stored in localstorage.
269
270   const contactName = localStorage.getItem('contact');
271
272   if (contactName === 'on') {
273     // Now I check for any saved contact
274
275   }
276
277   // I check localstorage for a saved contact form preference.
278   // If "on", I apply the form; otherwise, I don't.
279   // The better logic flows, updating both the class
280   // on the body and the icon (element).
281   // Your preference is stored in localstorage.
282
283   const contactName = localStorage.getItem('contact');
284
285   if (contactName === 'on') {
286     // Now I check for any saved contact
287
288   }
289
290   // I check localstorage for a saved contact form preference.
291   // If "on", I apply the form; otherwise, I don't.
292   // The better logic flows, updating both the class
293   // on the body and the icon (element).
294   // Your preference is stored in localstorage.
295
296   const contactName = localStorage.getItem('contact');
297
298   if (contactName === 'on') {
299     // Now I check for any saved contact
300
301   }
302
303   // I check localstorage for a saved contact form preference.
304   // If "on", I apply the form; otherwise, I don't.
305   // The better logic flows, updating both the class
306   // on the body and the icon (element).
307   // Your preference is stored in localstorage.
308
309   const contactName = localStorage.getItem('contact');
310
311   if (contactName === 'on') {
312     // Now I check for any saved contact
313
314   }
315
316   // I check localstorage for a saved contact form preference.
317   // If "on", I apply the form; otherwise, I don't.
318   // The better logic flows, updating both the class
319   // on the body and the icon (element).
320   // Your preference is stored in localstorage.
321
322   const contactName = localStorage.getItem('contact');
323
324   if (contactName === 'on') {
325     // Now I check for any saved contact
326
327   }
328
329   // I check localstorage for a saved contact form preference.
330   // If "on", I apply the form; otherwise, I don't.
331   // The better logic flows, updating both the class
332   // on the body and the icon (element).
333   // Your preference is stored in localstorage.
334
335   const contactName = localStorage.getItem('contact');
336
337   if (contactName === 'on') {
338     // Now I check for any saved contact
339
340   }
341
342   // I check localstorage for a saved contact form preference.
343   // If "on", I apply the form; otherwise, I don't.
344   // The better logic flows, updating both the class
345   // on the body and the icon (element).
346   // Your preference is stored in localstorage.
347
348   const contactName = localStorage.getItem('contact');
349
350   if (contactName === 'on') {
351     // Now I check for any saved contact
352
353   }
354
355   // I check localstorage for a saved contact form preference.
356   // If "on", I apply the form; otherwise, I don't.
357   // The better logic flows, updating both the class
358   // on the body and the icon (element).
359   // Your preference is stored in localstorage.
360
361   const contactName = localStorage.getItem('contact');
362
363   if (contactName === 'on') {
364     // Now I check for any saved contact
365
366   }
367
368   // I check localstorage for a saved contact form preference.
369   // If "on", I apply the form; otherwise, I don't.
370   // The better logic flows, updating both the class
371   // on the body and the icon (element).
372   // Your preference is stored in localstorage.
373
374   const contactName = localStorage.getItem('contact');
375
376   if (contactName === 'on') {
377     //
```

Começar o código, delimitando, assim, onde as tags vão funcionar e dessa forma organizando a aplicação.

<Head>: Cabeça do texto, tudo de

[illegible]

informações base,
ou seja, links de
outros arquivos,
fontes, ícones,
css, a parte da
linguagem do site,
título, etc...
parte lógica do
código html.</head>

<body>:Estrutura do
site que será
visível (ou não,
dependendo do
visibility:
hidden).</body>

Questão 6. Para que
serve a tag <title>
em uma página HTML?

<title> Define o
título da
aba/navegador</title>

Questão 7. Escreva o código mínimo para criar uma página HTML
com um título e um parágrafo.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
<!-- Minha dúvida era se o título era da página ou se era um
h1 -->
    <h1>Title</h1>
    <p>Starting the Web Application</p>
```

```
</body>
</html>
```

Questão 8. O que é o atributo alt da tag e por que ele é importante?

É importante para a acessibilidade - ou erro de conexão- , pois é um texto alternativo que aparece quando o usuário ou o site está com problemas de conexão, ou quando ele usa um leitor de tela.

Questão 9. Reproduza o código para exibir uma imagem com legenda usando as tags <figure> e <figcaption>.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <figure class="img-pedida">
      
      <figcaption>Legenda da imagem</figcaption>
    </figure>
  </body>
</html>
```

Questão 10. Quais são as diferenças entre e <i> em um texto HTML? Dê um exemplo de uso para cada.

: negrito. destacar texto por meio de acrescentar "peso" a fonte, deixando-a mais "preta, encorpada, aumentada".
<i>> itálico.</i> destacar texto por meio de inclinar a fonte.

Questão 11. Explique a diferença entre listas ordenadas () e não ordenadas (). Dê um exemplo de cada uma.

Ordenadas - possuem ordem e a ordem IMPORTA: Instruções, Passo a passo, manual.

Não Ordenadas - possuem até uma ordem, porém ela NÃO importa: Bullet list, To-do list, Lista de ingredientes.

Questão 12. Qual atributo da tag permite alterar o tipo de numeração de uma lista? Cite dois valores possíveis.

```
<ol type=""></ol>
```

O Type indica o tipo de numeração, isso facilita caso pegue uma lista fora de ordem, que precisa numerar.

Questão 13. Escreva um exemplo de link HTML que abra o site da UNIESP em uma nova aba.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <a href="https://www.iesp.edu.br/portal"
target="_blank">Nova Aba UNIESP</a>
  </body>
</html>
```

Questão 14. Qual a função da tag <form> e quais são os atributos mais importantes dessa tag?

<form> É realmente para criar um formulário, onde o cliente - usuário - pode editar os campos de texto - inputs - que venham a fornecer informações para a aplicação criada. Atributos: action | method | target | novalidate | autocomplete.

Questão 15. Cite pelo menos cinco tipos de campos de formulário (inputs) que podem ser utilizados no HTML e explique em que situações cada um pode ser aplicado.

text - mensagens, comentários.

password - senhas

email - email

date - data

number - números