



Criando seu próprio compilador JS

Descrição do projeto

Vamos criar nosso próprio compilador, estilizar, e hospedar na internet!

Para o processo de hospedagem, é necessário ter o github!

Serão utilizadas as tecnologias HTML, CSS E JavaScript!

Começando

Vamos criar um arquivo `index.html`, e utilizar o atalho (!) para criar nossa estrutura principal.



```
1 <div class="container">
2   <h1>Compilador de Código Javascript</h1> <!-- Título principal da página -->
3   <textarea id="codeInput" placeholder="Digite seu código aqui..."></textarea> <!-- Área de texto para entrada do código -->
4   <button onclick="compileCode()">Compilar</button> <!-- Botão para compilar o código -->
5   <div class="result" id="resultOutput"></div> <!-- Div para mostrar o resultado da compilação -->
6 </div>
```

Começando

Temos também um rodapé personalizado com as redes sociais e o nome do desenvolvedor, mas para pegar os ícones, do linkedin, github e Instagram, precisaremos de uma biblioteca chamada Fonte Awesome.

link do css do Fonte Awesome:

<https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css>

link do JavaScript do Fonte Awesome:

<https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/js/all.min.js>

Começando

Para adicionarmos o link do css da biblioteca, usaremos a tag link dentro do head:



```
1 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"> <!-- Inclui o CSS do Font Awesome -->
```

E para conectar o JavaScript, colocaremos uma tag script ao final do body:



```
1 <script src="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/js/all.min.js"></script> <!-- Inclui o JS do Font Awesome -->
```

Começando

Agora que conectamos a biblioteca, podemos criar nosso footer:

```
1 <footer><!-- Texto do rodapé -->
2   Desenvolvido por Nathalia Macedo<br>
3   <!-- Link para o LinkedIn com ícone -->
4   <a href="https://www.linkedin.com/in/nathalia-de-macedo-martins-4aa693253" target="_blank">
5     <i class="fab fa-linkedin"></i>
6   </a>
7   <!-- Link para o GitHub com ícone -->
8   <a href="https://github.com/Nathalia-Macedo" target="_blank">
9     <i class="fab fa-github"></i>
10  </a>
11
12  <!-- Link para o Instagram com ícone -->
13    <a href="https://www.instagram.com/nath_dev_/" target="_blank"><i class="fab fa-instagram"></i></a>
14  </footer>
```

Criando a parte lógica

Agora que nossa estrutura html está toda pronta, podemos criar as funcionalidades, mas antes, precisamos aprender a respeito da função `eval()`

A função `eval()` em JavaScript é uma função poderosa e potencialmente perigosa que avalia ou executa um código JavaScript representado como uma string. Aqui está uma explicação detalhada sobre como a função `eval()` funciona, seus usos e os cuidados que devemos ter ao usá-la.

Criando a parte lógica

A função `eval()` recebe uma string contendo código JavaScript como argumento e executa esse código dentro do escopo atual.

Exemplo:



```
1  const code = '2 + 2';  
2  const result = eval(code); // result será 4
```


Criando a parte lógica

Agora que aprendemos a função `eval()`, podemos criar o nosso código!



```
1  <script>
2    function compileCode() {
3      // Obtém o valor do código digitado
4      const code = document.getElementById('codeInput').value;
5      // Tenta executar o código
6      try {
7        eval(code);
8        // Mostra a mensagem de sucesso
9        document.getElementById('resultOutput').innerHTML = 'Código compilado com sucesso!';
10     // Mostra a mensagem de erro
11     } catch (error) {
12       document.getElementById('resultOutput').innerHTML = 'Erro ao compilar o código: ' + error.message;
13     }
14   }
15 </script>
```

Criando a estilização

Nosso código já está funcional, basta estilizar:



```
1  body {  
2      font-family: Arial, sans-serif; /* Define a fonte do corpo do texto */  
3      background-color: #000; /* Define a cor de fundo da página como preto */  
4      color: #fff; /* Define a cor do texto como branco */  
5      margin: 0; /* Remove a margem padrão do corpo */  
6      padding: 0; /* Remove o preenchimento padrão do corpo */  
7      display: flex; /* Usa flexbox para o layout */  
8      flex-direction: column; /* Define a direção dos itens flex como coluna */  
9      min-height: 100vh; /* Garante que o conteúdo ocupe pelo menos toda a altura da viewport */  
10 }
```

Criando a estilização

Nosso código já está funcional, basta estilizar:



```
1  .container {  
2      width: 90%; /* Ajusta para 90% da largura da tela */  
3      max-width: 800px; /* Define um máximo de 800px de largura */  
4      margin: auto; /* Centraliza horizontalmente */  
5      text-align: center; /* Centraliza o texto */  
6      padding-top: 50px; /* Adiciona espaçamento no topo */  
7      flex: 1; /* Ocupa o espaço restante verticalmente */  
8      display: flex; /* Usa flexbox para o layout */  
9      flex-direction: column; /* Define a direção dos itens flex como coluna */  
10     justify-content: center; /* Centraliza verticalmente o conteúdo */  
11 }
```


Criando a estilização


Nosso código já está funcional, basta estilizar:



```
1  textarea {
2      height: 200px; /* Define a altura da área de texto */
3      padding: 10px; /* Adiciona preenchimento interno */
4      margin-top: 10px; /* Adiciona margem no topo */
5      background-color: #333; /* Define a cor de fundo da área de texto */
6      color: #fff; /* Define a cor do texto da área de texto */
7      border: none; /* Remove a borda */
8      resize: none; /* Desabilita o redimensionamento da área de texto */
9      font-size: 16px; /* Define o tamanho da fonte */
10 }
```

Criando a estilização


Nosso código já está funcional, basta estilizar:



```
1  button {
2      background-color: #4CAF50; /* Define a cor de fundo do botão */
3      color: white; /* Define a cor do texto do botão */
4      padding: 14px 20px; /* Adiciona preenchimento interno */
5      margin: 8px 0; /* Adiciona margem superior e inferior */
6      border: none; /* Remove a borda */
7      cursor: pointer; /* Define o cursor como pointer (mãozinha) */
8      width: 100%; /* Define a largura do botão para 100% */
9      font-size: 16px; /* Define o tamanho da fonte */
10 }
```

Criando a estilização

Nosso código já está funcional, basta estilizar:



```
1  button:hover {
2      opacity: 0.8; /* Reduz a opacidade quando o botão é hover */
3  }
4  .result {
5      margin-top: 20px; /* Adiciona margem superior */
6      padding: 20px; /* Adiciona preenchimento interno */
7      background-color: #555; /* Define a cor de fundo */
8      color: #fff; /* Define a cor do texto no resultado */
9  }
10 footer {
11     text-align: center; /* Centraliza o texto */
12     margin-top: 20px; /* Adiciona margem superior */
13     color: #aaa; /* Define a cor do texto */
14     background-color: #111; /* Define a cor de fundo */
15     padding: 10px 0; /* Adiciona preenchimento superior e inferior */
16 }
```


Criando a estilização

Nosso código já está funcional, basta estilizar:



```
1 footer a {  
2     color: #aaa; /* Define a cor dos links */  
3     text-decoration: none; /* Remove a decoração do texto */  
4     margin: 0 10px; /* Adiciona margem horizontal aos links */  
5 }  
6 footer a:hover {  
7     color: #fff; /* Muda a cor dos links ao passar o mouse */  
8 }
```


Criando a estilização

Nosso código já está funcional, basta estilizar:



```
1  /* Estilos adicionais para telas menores */  
2  @media (max-width: 600px) {  
3      .container {  
4          width: 95%; /* Ajusta para 95% da largura da tela em telas menores */  
5      }  
6  }
```