

JS

JavaScript

Sumário

Introdução ao JavaScript	3
1. O que é JavaScript?	3
2. Configurando o Ambiente:	3
3. Primeiro Código: "Hello, World!"	3
Variáveis e Tipos de Dados	4
1. O que são variáveis?	4
2. Declarando Variáveis	4
3. Tipos de Dados em JavaScript	4
Operadores e Expressões	5
Estruturas de Controle	7
Funções e Escopo	8
Loops	11
Objetos	11
Manipulação do DOM (Document Object Model)	13
Eventos e Interatividade	14
Atividades	15

Introdução ao JavaScript

1. O que é JavaScript?

JavaScript é uma linguagem de programação que permite criar conteúdo interativo na web. Junto com HTML e CSS, ele é uma das três principais tecnologias da web. JavaScript é utilizado para:

1. Atualizar e manipular o conteúdo da página.
2. Controlar multimídia.
3. Criar animações e interações complexas.

2. Configurando o Ambiente:

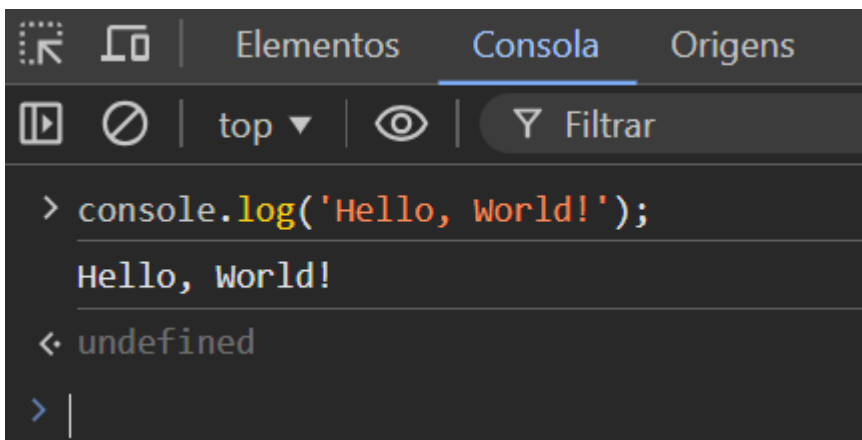
Utilizando o Console do Navegador

O console do navegador é uma ferramenta de depuração que permite testar código JavaScript diretamente em seu navegador. Acesse-o pressionando F12 ou Ctrl + Shift + J (no Google Chrome).

3. Primeiro Código: "Hello, World!"

Código no Console do Navegador:

Abra o console do navegador e digite:

A screenshot of a web browser's developer console. The console is open, showing the 'Console' tab. The input area at the top contains the code 'console.log('Hello, World!');'. Below the input, the output 'Hello, World!' is displayed. The console also shows 'undefined' as the return value of the log function. The interface includes icons for opening the console, a filter icon, and a search bar labeled 'Filtrar'.

Variáveis e Tipos de Dados

1. O que são variáveis?

Variáveis são "caixas" que armazenam dados. Elas têm um nome e um valor, e podem ser usadas para guardar e manipular informações ao longo do código.

Exemplo:

```
let nome = "Maria";  
let idade = 25;
```

Neste exemplo, nome é uma variável que armazena a string "Maria" e idade é uma variável que armazena o número 25.

2. Declarando Variáveis

var: Forma mais antiga de declarar variáveis. O escopo de var é global ou de função.

let: Introduzido no ES6, tem escopo de bloco e é a forma mais recomendada para variáveis mutáveis.

const: Também introduzido no ES6, é usado para variáveis que não devem ser reatribuídas (constantes).

```
var cidade = "São Paulo"; // Escopo global  
let ano = 2024; // Escopo de bloco  
const pi = 3.14159; // Constante
```

3. Tipos de Dados em JavaScript

Primitivos

- **String:** Cadeias de texto. Exemplo: "Olá, Mundo!"
- **Number:** Números inteiros e de ponto flutuante. Exemplo: 42 ou 3.14
- **Boolean:** Verdadeiro ou falso. Exemplo: true ou false
- **Null:** Representa a ausência intencional de valor. Exemplo: let x = null;
- **Undefined:** Valor não atribuído. Exemplo: let y;
- **Symbol:** Um valor único e imutável. Útil para identificar propriedades de objetos de forma única.

```
let nome = "Ana"; // String  
let idade = 30; // Number  
let casado = false; // Boolean  
let nada = null; // Null  
let indefinido; // Undefined  
let id = Symbol('id'); // Symbol
```

Compostos

- **Object:** Estruturas de dados que armazenam pares chave-valor.
- **Array:** Lista ordenada de valores, que podem ser de tipos diferentes.

Exemplos:

```
let pessoa = {  
  nome: "João",  
  idade: 35,  
  casado: true  
}; // Object  
  
let frutas = ["maçã", "banana", "laranja"]; // Array
```

Operadores e Expressões

1. O que são operadores?

Operadores são símbolos que dizem ao JavaScript para realizar uma ação específica em um ou mais valores. Eles podem ser usados para fazer cálculos, comparar valores, e realizar outras operações.

Exemplo:

```
let soma = 5 + 3; // O operador + realiza a adição de 5 e 3
```

2. Operadores Aritméticos

Os operadores aritméticos são usados para realizar operações matemáticas.

- **Adição (+)**
- **Subtração (-)**
- **Multiplicação (*)**
- **Divisão (/)**
- **Módulo (%):** Resto da divisão.

```
let a = 10;  
let b = 3;  
  
let soma = a + b; // 13  
let subtracao = a - b; // 7  
let multiplicacao = a * b; // 30  
let divisao = a / b; // 3.333...  
let modulo = a % b; // 1
```

Incremento e Decremento:

- **Incremento (++)**: Adiciona 1 ao valor da variável.
- **Decremento (--)**: Subtrai 1 do valor da variável.

```
let c = 5;  
c++; // c agora é 6  
c--; // c agora é 5 novamente
```

3. Operadores de Atribuição

Os operadores de atribuição são usados para atribuir valores a variáveis.

Atribuição Combinada

- **+=**: Adição e atribuição.
- **-=**: Subtração e atribuição.
- ***=**: Multiplicação e atribuição.
- **/=**: Divisão e atribuição.
- **%=**: Módulo e atribuição

```
let e = 15;  
e += 5; // e agora é 20  
e -= 2; // e agora é 18  
e *= 2; // e agora é 36  
e /= 6; // e agora é 6  
e %= 4; // e agora é 2
```

4. Operadores de Comparação

Os operadores de comparação são usados para comparar dois valores e retornar um booleano (`true` ou `false`).

- **Igualdade (==)**
- **Estritamente Igual (===)**
- **Desigualdade (!=)**
- **Estritamente Desigual (!==)**
- **Maior que (>)**
- **Menor que (<)**
- **Maior ou igual (>=)**
- **Menor ou igual (<=)**

```
let f = 10;  
let g = 20;  
  
console.log(f == g); // false  
console.log(f != g); // true  
console.log(f < g); // true  
console.log(f > g); // false  
console.log(f <= 10); // true  
console.log(f === "10"); // false (compara tipo e valor)
```

5. Operadores Lógicos

- **&& (AND):** Retorna `true` se ambas as expressões forem verdadeiras.
- **|| (OR):** Retorna `true` se pelo menos uma das expressões for verdadeira.
- **! (NOT):** Inverte o valor booleano.

```
let h = true;
let i = false;

console.log(h && i); // false
console.log(h || i); // true
console.log(!h); // false
```

6. Operadores de Concatenação

+: Concatena duas ou mais strings.

```
let primeiroNome = "João";
let ultimoNome = "Silva";

let nomeCompleto = primeiroNome + " " + ultimoNome; // "João Silva"
```

Estruturas de Controle

1. O que são Estruturas de Controle?

Estruturas de controle são comandos que permitem alterar o fluxo de execução do código com base em condições específicas. Elas são fundamentais para criar lógica condicional em programas.

```
let idade = 18;

if (idade >= 18) {
  console.log("Você é maior de idade.");
} else {
  console.log("Você é menor de idade.");
}
```

A estrutura `if` avalia uma condição e executa um bloco de código se a condição for verdadeira.

O `else` é usado para executar um bloco de código alternativo se a condição `if` for falsa.

O `else if` permite testar múltiplas condições em sequência:

```
let c = 7;

if (c > 10) {
    console.log("c é maior que 10");
} else if (c > 5) {
    console.log("c é maior que 5 mas menor ou igual a 10");
} else {
    console.log("c é menor ou igual a 5");
}
```

2. A Estrutura switch

O `switch` é uma estrutura de controle que executa diferentes blocos de código com base no valor de uma expressão.

```
let fruta = "maçã";

switch (fruta) {
    case "maçã":
        console.log("Você escolheu maçã.");
        break;
    case "banana":
        console.log("Você escolheu banana.");
        break;
    case "laranja":
        console.log("Você escolheu laranja.");
        break;
    default:
        console.log("Fruta não disponível.");
}
```

Funções e Escopo

1. O que são Funções?

Funções são blocos de código que realizam uma tarefa específica e podem ser reutilizados. Elas permitem organizar o código de forma modular e evitam a repetição de código.

```
function saudacao() {
    console.log("Olá, mundo!");
}
```

2. Criando Funções

A sintaxe básica para criar uma função envolve a palavra-chave `function`, seguida do nome da função, parênteses `()` e um bloco de código `{}`.

```
function dizerOla() {
    console.log("Olá!");
}
```


Parâmetros e Argumentos

Funções podem aceitar parâmetros, que são valores passados para a função quando ela é chamada.

```
function saudar(nome) {  
    console.log("Olá, " + nome + "!");  
}  
  
saudar("Carlos"); // Saída: Olá, Carlos!
```

Funções com Retorno

Funções podem retornar valores usando a palavra-chave `return`.

```
function soma(a, b) {  
    return a + b;  
}  
  
let resultado = soma(5, 3); // resultado é 8  
console.log(resultado);
```

3. Expressões de Função

Funções anônimas são funções sem nome, geralmente atribuídas a uma variável.

```
let saudacao = function() {  
    console.log("Olá!");  
};  
  
saudacao(); // Chama a função anônima
```

Arrow Functions (Funções de Seta)

Arrow functions são uma sintaxe mais curta para criar funções. Elas foram introduzidas no ES6.

```
let soma = (a, b) => a + b;  
  
console.log(soma(4, 6)); // Saída: 10
```

4. Escopo em JavaScript

Escopo Global

Variáveis declaradas fora de qualquer função têm escopo global, o que significa que podem ser acessadas de qualquer lugar no código.

```
let nome = "Ana"; // Escopo global

function mostrarNome() {
    console.log(nome); // Acessível dentro da função
}

mostrarNome();
```

Escopo de Função

Variáveis declaradas dentro de uma função têm escopo local (ou de função), e só podem ser acessadas dentro dessa função.

```
function exemplo() {
    let idade = 30; // Escopo de função
    console.log(idade);
}

exemplo();
// console.log(idade); // Erro: idade não está definida
```

Escopo de Bloco

O escopo de bloco foi introduzido com `let` e `const`. Variáveis declaradas dentro de um bloco `{ }` só podem ser acessadas dentro desse bloco.

```
if (true) {
    let cor = "azul"; // Escopo de bloco
    console.log(cor); // Acessível aqui
}

// console.log(cor); // Erro: cor não está definida
```

Loops

for Loop

O `for` loop é utilizado para repetir um bloco de código um número específico de vezes.

```
for (let i = 0; i < 5; i++) {  
  console.log(i);  
}  
// Saída: 0, 1, 2, 3, 4
```

while Loop

O `while` loop repete um bloco de código enquanto uma condição for verdadeira.

```
let contador = 0;  
  
while (contador < 5) {  
  console.log(contador);  
  contador++;  
}  
// Saída: 0, 1, 2, 3, 4
```

for...of Loop

O `for...of` loop é utilizado para iterar sobre elementos de um array.

```
let cores = ["vermelho", "verde", "azul"];  
  
for (let cor of cores) {  
  console.log(cor);  
}  
// Saída: vermelho, verde, azul
```

Objetos

1. O que são Objetos?

Objetos são coleções de pares chave-valor. Eles permitem agrupar dados relacionados e funcionalidades dentro de uma única estrutura. Em JavaScript, quase tudo é um objeto, incluindo arrays e funções.

Criando Objetos

A forma mais comum de criar um objeto em JavaScript é utilizando a sintaxe literal `{ }`.

```
let carro = {  
  marca: "Toyota",  
  modelo: "Corolla",  
  ano: 2020  
};
```

Acessando Propriedades

Você pode acessar as propriedades de um objeto utilizando a notação de ponto `.` ou colchetes `[]`.

```
console.log(carro.marca); // "Toyota"  
console.log(carro["modelo"]); // "Corolla"
```

Modificando Propriedades

As propriedades de um objeto podem ser modificadas ou adicionadas dinamicamente.

```
carro.ano = 2021; // Modificando uma propriedade existente  
carro.cor = "preto"; // Adicionando uma nova propriedade  
  
console.log(carro.ano); // 2021  
console.log(carro.cor); // "preto"
```

Métodos de Objetos

Métodos são funções que são propriedades de objetos. Eles podem ser usados para realizar ações com os dados do objeto.

```
let pessoa = {  
  nome: "Ana",  
  saudacao: function() {  
    console.log("Olá, " + this.nome + "!");  
  }  
};
```

Chamando Métodos

Você pode chamar um método de um objeto utilizando a notação de ponto.

```
pessoa.saudacao(); // Saída: "Olá, Ana!"
```

Iterando Sobre Objetos

O loop `for...in` é utilizado para iterar sobre as propriedades de um objeto.

```
for (let chave in pessoa) {  
    console.log(chave + ": " + pessoa[chave]);  
}  
// Saída:  
// nome: Ana  
// saudacao: function() { ... }
```

Métodos `Object.keys()` e `Object.values()`

Os métodos `Object.keys()` e `Object.values()` retornam, respectivamente, as chaves e os valores das propriedades de um objeto.

```
let chaves = Object.keys(carro);  
let valores = Object.values(carro);  
  
console.log(chaves); // ["marca", "modelo", "ano", "cor"]  
console.log(valores); // ["Toyota", "Corolla", 2021, "preto"]
```

Manipulação do DOM (Document Object Model)

O que é o DOM?

O DOM (Document Object Model) é uma representação em árvore da estrutura de um documento HTML ou XML. Ele permite que JavaScript acesse e manipule o conteúdo e a estrutura de páginas web.

Selecionando Elementos do DOM

`getElementById()`

adicione um `id= 'titulo'` em uma tag html e teste o código no console

```
let titulo = document.getElementById("titulo");  
console.log(titulo.innerText); // Exibe o texto do elemento com id "titulo"
```

Modificando Conteúdo e Estilo dos Elementos

Use a propriedade `innerText` ou `textContent` para alterar o texto de um elemento.

```
let titulo = document.getElementById("titulo");  
titulo.innerText = "Novo Título"; // Altera o texto do elemento
```

Alterando o HTML

Use a propriedade `innerHTML` para alterar o HTML interno de um elemento.

```
let conteudo = document.getElementById("conteudo");
conteudo.innerHTML = "<p>Novo parágrafo.</p>"; // Substitui o conteúdo do elemento
```

Alterando Estilos com style

Use a propriedade `style` para alterar o estilo CSS de um elemento.

```
let subtítulo = document.getElementById("subtítulo");
subtítulo.style.color = "blue"; // Altera a cor do texto para azul
subtítulo.style.fontSize = "20px"; // Altera o tamanho da fonte
```

Eventos e Interatividade

O que são Eventos?

Eventos são ações ou ocorrências que acontecem no navegador, como cliques de botão, movimentos do mouse, ou teclas pressionadas. JavaScript pode detectar esses eventos e responder a eles, permitindo interações dinâmicas.

Exemplo: Um clique de botão dispara um evento de clique, que pode ser capturado para executar uma função. Veja a seguir nas atividades

Atividades

Atividade 1: To-Do List Dinâmica

Criar uma lista de tarefas onde o usuário pode adicionar, remover e marcar tarefas como concluídas.

```
1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>To-Do List</title>
7   <style>
8     /* Estilo para tarefas concluídas */
9     .concluida {
10       text-decoration: line-through;
11       color: grey;
12     }
13   </style>
14 </head>
15 <body>
16   <h1>Minha Lista de Tarefas</h1>
17   <!-- Formulário para adicionar novas tarefas -->
18   <form id="formTarefa">
19     <input type="text" id="novaTarefa" placeholder="Adicionar nova tarefa">
20     <button type="submit">Adicionar</button>
21   </form>
22   <!-- Lista onde as tarefas serão exibidas -->
23   <ul id="listaTarefas"></ul>
24
25   <script src="index.js"></script>
26 </body>
27 </html>
```

```
JS > atividades > to_do_list > JS index.js > ...
1 let form = document.getElementById("formTarefa");
2 let listaTarefas = document.getElementById("listaTarefas");
3
4 // Evento para adicionar uma nova tarefa
5 form.addEventListener("submit", function(evento) {
6   evento.preventDefault(); // Previne o comportamento padrão do formulário
7   let tarefaTexto = document.getElementById("novaTarefa").value;
8
9   // Verifica se o campo de texto não está vazio
10  if (tarefaTexto !== "") {
11    // Cria um novo item de lista (li)
12    let novaTarefa = document.createElement("li");
13    novaTarefa.innerText = tarefaTexto;
14
15    // Adiciona evento para marcar a tarefa como concluída
16    novaTarefa.addEventListener("click", function() {
17      this.classList.toggle("concluida");
18    });
19
20    // Cria um botão para remover a tarefa
21    let botaoRemover = document.createElement("button");
22    botaoRemover.innerText = "Remover";
23    botaoRemover.addEventListener("click", function() {
24      listaTarefas.removeChild(novaTarefa); // Remove a tarefa da lista
25    });
26
27    // Adiciona o botão de remover à nova tarefa e a tarefa à lista
28    novaTarefa.appendChild(botaoRemover);
29    listaTarefas.appendChild(novaTarefa);
30    document.getElementById("novaTarefa").value = ""; // Limpa o campo de texto
31  }
32 });
```

Atividade 2: Jogo de Adivinhação de Números

Desenvolver um jogo onde o usuário precisa adivinhar um número gerado aleatoriamente pelo sistema.

```
JS > atividades > jogo_adv > index.html > ...
1  <!DOCTYPE html>
2  <html lang="pt-BR">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Jogo de Adivinhação</title>
7      <style>
8          #resultado {
9              margin-top: 20px;
10             font-weight: bold;
11         }
12     </style>
13 </head>
14 <body>
15     <h1>Jogo de Adivinhação</h1>
16     <p>Tente adivinhar o número entre 1 e 100!</p>
17     <!-- Campo de entrada para o palpite do usuário -->
18     <input type="number" id="palpite" min="1" max="100">
19     <button id="adivinhar">Adivinhar</button>
20     <!-- Área para exibir o resultado e a contagem de tentativas -->
21     <p id="resultado"></p>
22     <p id="tentativas">Tentativas: 0</p>
23     <button id="reiniciar">Reiniciar</button>
24
25 <script src="index.js"></script>
26 </body>
27 </html>
28
```

```
1  let numeroAleatorio = Math.floor(Math.random() * 100) + 1; // Gera um número aleatório entre 1 e 100
2  let tentativas = 0;
3
4  // Evento para processar o palpite do usuário
5  document.getElementById("adivinhar").addEventListener("click", function() {
6      let palpite = parseInt(document.getElementById("palpite").value);
7      tentativas++; // Incrementa o número de tentativas
8      let resultado = document.getElementById("resultado");
9
10     // Verifica se o palpite está correto, muito alto ou muito baixo
11     if (palpite === numeroAleatorio) {
12         resultado.innerHTML = "Parabéns! Você adivinhou o número!";
13         resultado.style.color = "green";
14     } else if (palpite > numeroAleatorio) {
15         resultado.innerHTML = "Muito alto! Tente novamente.";
16         resultado.style.color = "red";
17     } else {
18         resultado.innerHTML = "Muito baixo! Tente novamente.";
19         resultado.style.color = "red";
20     }
21
22     // Atualiza o contador de tentativas
23     document.getElementById("tentativas").innerHTML = "Tentativas: " + tentativas;
24 });
25
26 // Evento para reiniciar o jogo
27 document.getElementById("reiniciar").addEventListener("click", function() {
28     numeroAleatorio = Math.floor(Math.random() * 100) + 1; // Gera um novo número aleatório
29     tentativas = 0; // Reseta o contador de tentativas
30     document.getElementById("resultado").innerHTML = ""; // Limpa a mensagem de resultado
31     document.getElementById("tentativas").innerHTML = "Tentativas: 0"; // Reseta a exibição de tentativas
32     document.getElementById("palpite").value = ""; // Limpa o campo de entrada
33 });
```


Atividade 3: Quiz Interativo

Desenvolver um quiz interativo onde o usuário pode responder a perguntas e ver seu resultado final.

```
1  <!DOCTYPE html>
2  <html lang="pt-BR">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Quiz Interativo</title>
7      <style>
8          .pergunta {
9              margin-bottom: 20px;
10         }
11         .resultado {
12             margin-top: 20px;
13             font-weight: bold;
14         }
15     </style>
16 </head>
17 <body>
18     <h1>Quiz de JavaScript</h1>
19     <!-- Formulário do quiz -->
20     <form id="quizForm">
21         <!-- Pergunta 1 -->
22         <div class="pergunta">
23             <p>1. Qual a linguagem usada para criar interatividade em páginas web?</p>
24             <input type="radio" name="p1" value="HTML"> HTML<br>
25             <input type="radio" name="p1" value="CSS"> CSS<br>
26             <input type="radio" name="p1" value="JavaScript"> JavaScript
27         </div>
28         <!-- Pergunta 2 -->
29         <div class="pergunta">
30             <p>2. Qual tag é usada para criar um parágrafo em HTML?</p>
31             <input type="radio" name="p2" value="div"> div<br>
32             <input type="radio" name="p2" value="p"> p<br>
33             <input type="radio" name="p2" value="h1"> h1
34         </div>
35         <!-- Pergunta 3 -->
36         <div class="pergunta">
37             <p>3. Qual propriedade CSS é usada para mudar a cor de fundo?</p>
38             <input type="radio" name="p3" value="color"> color<br>
39             <input type="radio" name="p3" value="background-color"> background-color<br>
40             <input type="radio" name="p3" value="font-size"> font-size
41         </div>
42         <!-- Botão para enviar as respostas -->
43         <button type="button" onclick="verificarRespostas()">Enviar Respostas</button>
44     </form>
45     <!-- Área para exibir o resultado do quiz -->
46     <div id="resultado" class="resultado"></div>
47
48     <!--Link JS-->
49     <script src="index.js"></script>
50 </body>
51 </html>
```

```

1 // Objeto que armazena as respostas corretas
2 let respostasCorretas = {
3   p1: "JavaScript",
4   p2: "p",
5   p3: "background-color"
6 };
7 // Função para verificar as respostas do usuário
8 function verificarRespostas() {
9   let formulario = document.getElementById("quizForm");
10  let pontuacao = 0;
11  // Loop através das respostas corretas
12  for (let pergunta in respostasCorretas) {
13    // Verifica a resposta do usuário para cada pergunta
14    let respostaSelecionada = formulario.elements[pergunta].value;
15    if (respostaSelecionada === respostasCorretas[pergunta]) {
16      pontuacao++; // Incrementa a pontuação se a resposta estiver correta
17    }
18  }
19  // Exibe a pontuação final do usuário
20  let resultado = document.getElementById("resultado");
21  resultado.innerText = "Você acertou " + pontuacao + " de 3 perguntas.";
22 }

```

Atividade 4: Formulário Dinâmico com Validação

Neste exemplo, criamos um formulário de cadastro simples com validação de campos e manipulação de DOM. Ao enviar o formulário, verificamos se todos os campos estão preenchidos corretamente. Se houver erros, eles são destacados, e uma mensagem é exibida.

```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Cadastro de Usuário</title>
7   <style>
8     .erro {
9       border: 2px solid red;
10    }
11    .mensagem-erro {
12      color: red;
13      font-size: 14px;
14    }
15  </style>
16 </head>
17 <body>
18   <h1>Cadastro de Usuário</h1>
19   <form id="formCadastro">
20     <!-- Campos do formulário -->
21     <div>
22       <label for="nome">Nome:</label>
23       <input type="text" id="nome">
24       <p id="erroNome" class="mensagem-erro"></p>
25     </div>
26     <div>
27       <label for="email">Email:</label>
28       <input type="email" id="email">
29       <p id="erroEmail" class="mensagem-erro"></p>
30     </div>
31     <div>
32       <label for="senha">Senha:</label>
33       <input type="password" id="senha">
34       <p id="erroSenha" class="mensagem-erro"></p>
35     </div>
36     <button type="submit">Cadastrar</button>
37   </form>
38   <p id="mensagemFinal"></p>
39
40   <script src="index.js"></script>
41 </body>
42 </html>

```

```

1 // Referências aos elementos do DOM
2 let form = document.getElementById("formCadastro");
3 let nome = document.getElementById("nome");
4 let email = document.getElementById("email");
5 let senha = document.getElementById("senha");
6 let mensagemFinal = document.getElementById("mensagemFinal");
7
8 // Evento de submissão do formulário
9 form.addEventListener("submit", function(evento) {
10     evento.preventDefault(); // Impede o envio do formulário
11
12     let formValido = true;
13
14     // Validação do campo nome
15     if (nome.value === "") {
16         nome.classList.add("erro");
17         document.getElementById("erroNome").innerText = "O nome é obrigatório.";
18         formValido = false;
19     } else {
20         nome.classList.remove("erro");
21         document.getElementById("erroNome").innerText = "";
22     }
23
24     // Validação do campo email
25     if (email.value === "") {
26         email.classList.add("erro");
27         document.getElementById("erroEmail").innerText = "O email é obrigatório.";
28         formValido = false;
29     } else {
30         email.classList.remove("erro");
31         document.getElementById("erroEmail").innerText = "";
32     }
33
34     // Validação do campo senha
35     if (senha.value.length < 6) {
36         senha.classList.add("erro");
37         document.getElementById("erroSenha").innerText = "A senha deve ter pelo menos 6 caracteres.";
38         formValido = false;
39     } else {
40         senha.classList.remove("erro");
41         document.getElementById("erroSenha").innerText = "";
42     }
43
44     // Se o formulário for válido, exibe uma mensagem de sucesso
45     if (formValido) {
46         mensagemFinal.innerText = "Cadastro realizado com sucesso!";
47         mensagemFinal.style.color = "green";
48     } else {
49         mensagemFinal.innerText = "Por favor, corrija os erros acima.";
50         mensagemFinal.style.color = "red";
51     }
52 });

```