E-BOOK

JavaScript

JS

BÁSICO

Projetos Práticos

PROGRAMATIZANDO

CODIFIQUE SEU MUNDO!

INÍCIO

JAVASCRIPT BÁSICO

Capítulo 1: Introdução ao JavaScript

- O que é JavaScript e sua importância.
- Sintaxe básica do JavaScript: variáveis, tipos de dados, operadores.
- Incorporando JavaScript em um documento HTML.
- Saída de dados: console.log() e alert().
- Primeiros passos com interatividade.

Capítulo 2: Estruturas de Controle

- Condições: if, else if, else.
- Loops: for, while, do while.
- Switch case.
- Uso de operadores lógicos.
- Exemplos práticos de estruturas de controle.

Capítulo 3: Funções e Escopo

- O que são funções e por que são úteis.
- Declaração e chamada de funções.
- Parâmetros e argumentos de função.
- Escopo de variáveis: global vs. local.
- Exemplos de funções simples e reutilizáveis.

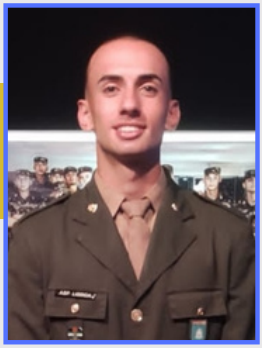
Capítulo 4: Manipulação do DOM

- O que é o DOM (Modelo de Objeto de Documento) e sua importância.
- Selecionando elementos do DOM.
- Manipulando conteúdo, estilo e atributos.
- Criando e removendo elementos dinamicamente.
- Exemplos práticos de manipulação do DOM.

Capítulo 5: Eventos e Event Listeners

- O que são eventos e como funcionam.
- Adicionando event listeners a elementos HTML.
- Tipos comuns de eventos: clique, mouseover, submit, etc.
- Manipulando eventos: prevenindo o comportamento padrão, propagando eventos.
- Exemplos práticos de interação do usuário com eventos.

AUTOR



Quem sou?

Olá! Sou Carlos Eduardo, ex-militar do Exército Brasileiro e agora mergulhando no mundo da engenharia de software. No meu dia a dia, lidero o setor de T.I. em uma distribuidora de materiais de segurança do trabalho.

Estou aqui para te contar algo emocionante! Preparei um material exclusivo para aqueles que estão começando no mundo do HTML. Você aprenderá a criar sites simples e ganhará uma sólida base no Front-end, com alguns truques de CSS e JavaScript para dar aquele toque especial.

E tem mais! Ao concluir o curso, você terá acesso a descontos especiais em nossos combos e outras apostilas. Não perca essa oportunidade de ampliar seus conhecimentos e economizar ao mesmo tempo. Estou ansioso para ver você se destacando neste universo da programação! Vamos nessa?

CAPITULO: 1

INTRODUÇÃO AO JAVASCRIPT

INTRODUÇÃO AO JAVASCRIPT

JAVASCRIPT É UMA LINGUAGEM DE PROGRAMAÇÃO AMPLAMENTE UTILIZADA PARA DESENVOLVIMENTO WEB. ENQUANTO HTML DÁ À SUA PÁGINA ESTRUTURA E CSS CUIDA DO ESTILO, JAVASCRIPT É A FERRAMENTA QUE ADICIONA INTERATIVIDADE E DINAMISMO.

Ele é executado no lado do cliente, ou seja, no navegador do usuário, permitindo que as páginas web reajam às ações dos usuários, atualizem conteúdos dinamicamente e forneçam uma experiência mais envolvente. Sua importância reside na capacidade de criar aplicativos web ricos e interativos, desde simples formulários até complexas aplicações de uma única página (SPA).

Sintaxe básica do JavaScript: variáveis, tipos de dados, operadores:

- Variáveis: Em JavaScript, você pode declarar variáveis usando as palavras-chave var, let ou const. Exemplo:

```
let nome = "João";
```

- Tipos de dados: JavaScript é uma linguagem dinamicamente tipada, o que significa que não é necessário declarar o tipo de uma variável ao criá-la. Os tipos de dados incluem strings, números, booleanos, arrays, objetos, entre outros.
- Operadores: JavaScript suporta diversos operadores, como aritméticos (+, -, *, /), de atribuição (=), de comparação (==, !=, ===, !==), lógicos (&&, ||, !), entre outros.

Incorporando JavaScript em um documento HTML:

JavaScript pode ser incorporado diretamente em um documento HTML usando a tag <script>. Isso permite que o código JavaScript seja executado quando a página é carregada. Exemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo de JavaScript</title>
</head>
<body>
    <script>
        // Seu código JavaScript aqui
    </script>
</body>
</html>
```

Saída de dados: console.log() e alert():

console.log(): É usado para exibir dados no console do navegador. É útil para depuração e registro de informações. Exemplo:

```
console.log("Olá, mundo!");
```

alert(): Exibe um diálogo com uma mensagem para o usuário. É útil para alertas simples, mas pode ser intrusivo se usado em excesso. Exemplo:

```
alert("Seu cadastro foi realizado com sucesso!");
```

Primeiros passos com interatividade: Para adicionar interatividade em uma página web, você pode usar eventos, que são acionados quando ocorrem determinadas ações do usuário, como clicar em um botão ou mover o mouse. Por exemplo, você pode adicionar um evento de clique a um botão para executar uma função JavaScript quando o botão for clicado:

```
<!DOCTYPE html>
<html>
<head>
    <title>Interatividade com JavaScript</title>
</head>
<body>
    <button onclick="mostrarMensagem()">Clique Aqui</button>
    <script>
        function mostrarMensagem() {
            alert("Você clicou no botão!");
        }
    </script>
</body>
</html>
```

ANOTAÇÕES:

CAPITULO: 2

ESTRUTURAS DE CONTROLE

ESTRUTURAS DE CONTROLE

AS ESTRUTURAS CONDICIONAIS PERMITEM QUE SEU CÓDIGO TOME DECISÕES COM BASE EM DIFERENTES CONDIÇÕES.

- if: Executa um bloco de código se uma condição for verdadeira.
- else if: Adiciona condições adicionais a um bloco if.
- else: Executa um bloco de código se todas as condições anteriores forem falsas.

Exemplo:

```
let hora = 20;

if (hora < 12) {
    console.log("Bom dia!");
} else if (hora < 18) {
    console.log("Boa tarde!");
} else {
    console.log("Boa noite!");
}
```

- Loops: for, while, do while
- Os loops permitem que você execute um bloco de código repetidamente, com base em uma condição.
- for: Usado quando você sabe quantas vezes quer repetir um bloco de código.

```
for (let i = 0; i < 5; i++) {
    console.log("O valor de i é " + i);
}
```

while: Executa um bloco de código enquanto uma condição é verdadeira.

```
let j = 0;
while (j < 5) {
    console.log("O valor de j é " + j);
    j++;
}
```

do while: Similar ao while, mas garante que o bloco de código seja executado pelo menos uma vez.

- Loops: for, while, do while
- Os loops permitem que você execute um bloco de código repetidamente, com base em uma condição.
- for: Usado quando você sabe quantas vezes quer repetir um bloco de código.

```
let k = 0;
do {
    console.log("O valor de k é " + k);
    k++;
} while (k < 5);
```

Switch case
O switch case é usado para executar um bloco de código entre várias opções com base no valor de uma expressão.
Exemplo:

```
let diaDaSemana = 3;

switch (diaDaSemana) {
    case 1:
        console.log("Segunda-feira");
        break;
    case 2:
        console.log("Terça-feira");
        break;
    case 3:
        console.log("Quarta-feira");
        break;
    case 4:
        console.log("Quinta-feira");
        break;
    case 5:
        console.log("Sexta-feira");
        break;
    case 6:
        console.log("Sábado");
        break;
    case 7:
        console.log("Domingo");
        break;
    default:
        console.log("Dia inválido");
}
```

Uso de operadores lógicos

- Operadores lógicos permitem combinar ou inverter condições.
- && (AND): Verdadeiro se ambas as condições forem verdadeiras.
- || (OR): Verdadeiro se pelo menos uma condição for verdadeira.
- ! (NOT): Inverte o valor lógico de uma condição.

Exemplo:

```
let idade = 25;
let possuiCarteiraDeMotorista = true;

if (idade >= 18 && possuiCarteiraDeMotorista) {
    console.log("Você pode dirigir.");
} else {
    console.log("Você não pode dirigir.");
}
```

Exemplos práticos de estruturas de controle

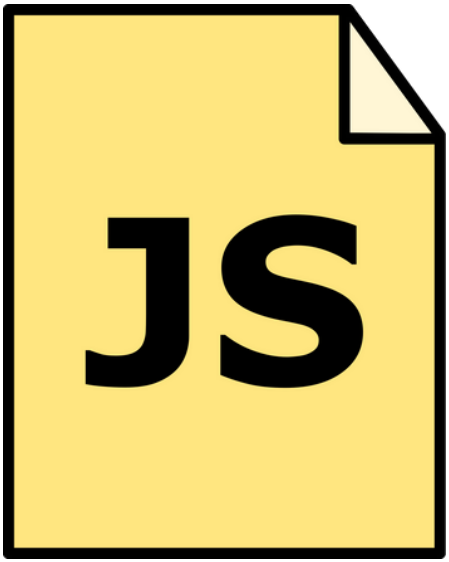
Vamos combinar várias estruturas de controle em um exemplo prático. Suponha que você está criando um sistema simples de autenticação de usuário:

```
let usuario = "admin";
let senha = "1234";

function autenticar(usuarioInput, senhaInput) {
    if (usuarioInput === usuario && senhaInput === senha) {
        console.log("Autenticação bem-sucedida!");
    } else {
        console.log("Nome de usuário ou senha incorretos.");
    }
}

autenticar("admin", "1234"); // Autenticação bem-sucedida!
autenticar("user", "1234");  // Nome de usuário ou senha incorretos.
```

Neste exemplo, a função autenticar usa uma condição if para verificar se o nome de usuário e a senha fornecidos correspondem aos valores esperados. Se ambos forem verdadeiros, a autenticação é bem-sucedida; caso contrário, uma mensagem de erro é exibida.



ANOTAÇÕES:

CAPÍTULO: 3

FUNÇÕES E ESCOPO

FUNÇÕES E ESCOPO

O QUE SÃO FUNÇÕES E POR QUE SÃO ÚTEIS

- Funções são blocos de código reutilizáveis que realizam uma tarefa específica. Elas ajudam a organizar e estruturar seu código, tornando-o mais modular e fácil de entender. Além disso, as funções permitem evitar a duplicação de código, facilitam a manutenção e melhoram a legibilidade.
- Declaração e chamada de funções
- Para declarar uma função, você usa a palavra-chave function, seguida pelo nome da função, parênteses e um bloco de código entre chaves. Para chamar uma função, você usa seu nome seguido de parênteses.

Exemplo:

```
// Declaração de função
function saudar() {
    console.log("Olá, mundo!");
}

// Chamada de função
saudar(); // Output: Olá, mundo!
```

- Parâmetros e argumentos de função
- Parâmetros são variáveis que você define na declaração da função para receber valores quando a função é chamada. Argumentos são os valores reais passados para a função quando ela é chamada.

Exemplo:

```
// Função com parâmetros
function somar(a, b) {
    return a + b;
}

// Chamando a função com argumentos
let resultado = somar(5, 3);
console.log(resultado); // Output: 8
```

- Escopo de variáveis: global vs. local
- O escopo de uma variável determina onde ela pode ser acessada no código.
- Global: Variáveis declaradas fora de qualquer função têm escopo global e podem ser acessadas de qualquer lugar no script.
- Local: Variáveis declaradas dentro de uma função têm escopo local e só podem ser acessadas dentro dessa função.

Exemplo:

```
let variavelGlobal = "Eu sou global";

function minhaFuncao() {
    let variavelLocal = "Eu sou local";
    console.log(variavelGlobal); // Acessível
    console.log(variavelLocal);  // Acessível
}

minhaFuncao();
console.log(variavelGlobal); // Acessível
// console.log(variavelLocal); // Erro: variavelLocal não está definida
```

- Exemplos de funções simples e reutilizáveis
- Aqui estão alguns exemplos de funções simples que podem ser reutilizadas em diferentes partes de um programa.

- Exemplo 1: Função para verificar se um número é par ou ímpar

```
function ehPar(numero) {
    return numero % 2 === 0;
}

console.log(ehPar(4)); // Output: true
console.log(ehPar(7)); // Output: false
```

Exemplo 2: Função para calcular a área de um círculo

```
function calcularAreaCirculo(raio) {
    const pi = 3.14159;
    return pi * raio * raio;
}

let area = calcularAreaCirculo(5);
console.log(area); // Output: 78.53975
```

Exemplo 3: Função para converter temperaturas de Celsius para Fahrenheit

```
function converterCelsiusParaFahrenheit(celsius) {
    return (celsius * 9/5) + 32;
}

let temperaturaF = converterCelsiusParaFahrenheit(25);
console.log(temperaturaF); // Output: 77
```

Esses exemplos mostram como as funções podem ser utilizadas para encapsular lógica específica, facilitando a reutilização e a manutenção do código. Além disso, entender o escopo das variáveis ajuda a evitar erros e a manter o código limpo e organizado.

ANOTAÇÕES:

CAPÍTULO: 4

MANIPULAÇÃO DO DOM

MANIPULAÇÃO DO DOM

O QUE É O DOM (MODELO DE OBJETO DE DOCUMENTO) E SUA IMPORTÂNCIA

O DOM (Document Object Model) é uma interface de programação para documentos HTML e XML. Ele representa a estrutura de um documento como uma árvore de objetos, onde cada nodo é um elemento, atributo ou texto. O DOM permite que linguagens de programação, como JavaScript, manipulem a estrutura, o estilo e o conteúdo de um documento de forma dinâmica.
Essa capacidade de modificar a página web em tempo real é crucial para criar interfaces de usuário interativas e dinâmicas.
Selecionando elementos do DOM

Para manipular elementos do DOM, primeiro é necessário selecioná-los. JavaScript oferece várias maneiras de fazer isso:

- getElementById: Seleciona um elemento pelo seu ID.
- getElementsByClassName: Seleciona todos os elementos com uma determinada classe.

- getElementsByTagName: Seleciona todos os elementos com um determinado nome de tag.
- querySelector: Seleciona o primeiro elemento que corresponde a um seletor CSS.
- querySelectorAll: Seleciona todos os elementos que correspondem a um seletor CSS.

Exemplo:

```
let elementoPorId = document.getElementById('meuId');
let elementosPorClasse = document.getElementsByClassName('minhaClasse');
let elementosPorTag = document.getElementsByTagName('p');
let elementoPorSeletor = document.querySelector('.minhaClasse');
let todosElementosPorSeletor = document.querySelectorAll('div');
```

- Manipulando conteúdo, estilo e atributos
- Depois de selecionar um elemento, você pode modificar seu conteúdo, estilo e atributos.
- Conteúdo: Usando innerHTML ou textContent.
- Estilo: Através da propriedade style.
- Atributos: Com métodos como setAttribute e getAttribute.

Exemplo:

```
let elemento = document.getElementById('meuId');

// Alterando o conteúdo
elemento.innerHTML = 'Novo conteúdo';
elemento.textContent = 'Novo texto';

// Alterando o estilo
elemento.style.color = 'blue';
elemento.style.backgroundColor = 'yellow';

// Manipulando atributos
elemento.setAttribute('data-custom', 'valor');
let atributo = elemento.getAttribute('data-custom');
console.log(atributo); // Output: valor
```

- Criando e removendo elementos dinamicamente
- Você pode criar novos elementos e adicioná-los ao DOM, bem como remover elementos existentes.
- Criando elementos: Usando createElement e appendChild ou insertBefore.
- Removendo elementos: Usando removeChild.
- Exemplo:

```
// Criando um novo elemento
let novoElemento = document.createElement('div');
novoElemento.textContent = 'Eu sou um novo elemento';

// Adicionando o novo elemento ao corpo do documento
document.body.appendChild(novoElemento);

// Removendo um elemento existente
let elementoParaRemover = document.getElementById('meuId');
elementoParaRemover.parentNode.removeChild(elementoParaRemover);
```

- Exemplos práticos de manipulação do DOM
- Aqui estão alguns exemplos práticos que demonstram como manipular o DOM de forma eficaz:
- Exemplo 1: Alterar o texto de um parágrafo ao clicar em um botão

```html
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo de Manipulação do DOM</title>
</head>
<body>
    <p id="meuParagrafo">Texto original</p>
    <button onclick="alterarTexto()">Clique aqui</button>

    <script>
        function alterarTexto() {
            let paragrafo = document.getElementById('meuParagrafo');
            paragrafo.textContent = 'Texto alterado!';
        }
    </script>
</body>
</html>
```

Exemplo 2: Adicionar um item a uma lista ao clicar em um botão

```html
<!DOCTYPE html>
<html>
<head>
    <title>Adicionar Item à Lista</title>
</head>
<body>
    <ul id="minhaLista">
        <li>Item 1</li>
        <li>Item 2</li>
    </ul>
    <button onclick="adicionarItem()">Adicionar Item</button>

    <script>
        function adicionarItem() {
            let lista = document.getElementById('minhaLista');
            let novoItem = document.createElement('li');
            novoItem.textContent = 'Novo Item';
            lista.appendChild(novoItem);
        }
    </script>
</body>
</html>
```

Exemplo 3: Alternar a visibilidade de um elemento

```
<!DOCTYPE html>
<html>
<head>
    <title>Alternar Visibilidade</title>
</head>
<body>
    <div id="meuDiv">Este é um div que pode ser ocultado ou mostrado.</div>
    <button onclick="alternarVisibilidade()">Alternar Visibilidade</button>

    <script>
        function alternarVisibilidade() {
            let div = document.getElementById('meuDiv');
            if (div.style.display === 'none') {
                div.style.display = 'block';
            } else {
                div.style.display = 'none';
            }
        }
    </script>
</body>
</html>
```

ANOTAÇÕES:

CAPÍTULO: 5

EVENTOS E EVENT LISTENERS

EVENTOS E EVENT LISTENERS

O QUE SÃO EVENTOS E COMO FUNCIONAM

Eventos são ações ou ocorrências que acontecem no navegador, como cliques do mouse, pressionamento de teclas, carregamento de páginas, entre outros. Eles permitem que você interaja com os usuários e responda a suas ações. Os eventos são fundamentais para criar interfaces dinâmicas e interativas, pois permitem que o JavaScript "ouça" e reaja às interações dos usuários.

Adicionando event listeners a elementos HTML
Um event listener é uma função que espera por um evento específico e executa uma ação quando esse evento ocorre. Para adicionar um event listener a um elemento HTML, você usa o método addEventListener.
Exemplo:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Event Listener</title>
</head>
<body>
    <button id="meuBotao">Clique aqui</button>

    <script>
        let botao = document.getElementById('meuBotao');
        botao.addEventListener('click', function() {
            alert('Botão clicado!');
        });
    </script>
</body>
</html>
```

Tipos comuns de eventos: clique, mouseover, submit, etc.

Existem muitos tipos de eventos que você pode usar para interagir com os usuários. Aqui estão alguns dos mais comuns:

- click: Disparado quando um elemento é clicado.
- mouseover: Disparado quando o mouse passa sobre um elemento.
- mouseout: Disparado quando o mouse sai de um elemento.
- submit: Disparado quando um formulário é enviado.
- keydown: Disparado quando uma tecla é pressionada.
- keyup: Disparado quando uma tecla é solta.

Exemplo:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Tipos de Eventos</title>
</head>
<body>
    <button id="meuBotao">Clique aqui</button>
    <input id="meuInput" type="text" placeholder="Digite algo">

    <script>
        let botao = document.getElementById('meuBotao');
        let input = document.getElementById('meuInput');

        botao.addEventListener('click', function() {
            console.log('Botão clicado!');
        });

        input.addEventListener('keydown', function(event) {
            console.log('Tecla pressionada: ' + event.key);
        });

        input.addEventListener('mouseover', function() {
            console.log('Mouse sobre o input!');
        });
    </script>
</body>
</html>
```

Manipulando eventos: prevenindo o comportamento padrão, propagando eventos

Às vezes, você pode querer impedir o comportamento padrão de um evento (por exemplo, impedir que um link seja seguido ou que um formulário seja enviado). Você pode fazer isso usando o método preventDefault.

Você também pode controlar a propagação de eventos na árvore do DOM usando stopPropagation. Exemplo:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Manipulando Eventos</title>
</head>
<body>
    <a href="https://www.example.com" id="meuLink">Não clique aqui</a>
    <div id="pai">
        <button id="meuBotao">Clique aqui</button>
    </div>

    <script>
        let link = document.getElementById('meuLink');
        let botao = document.getElementById('meuBotao');
        let pai = document.getElementById('pai');

        link.addEventListener('click', function(event) {
            event.preventDefault();
            alert('O comportamento padrão foi prevenido!');
        });

        botao.addEventListener('click', function(event) {
            event.stopPropagation();
            alert('Botão clicado, propagação parada!');
        });

        pai.addEventListener('click', function() {
            alert('Div pai clicado!');
        });
    </script>
</body>
</html>
```

Exemplos práticos de interação do usuário com eventos

Aqui estão alguns exemplos práticos de como os eventos podem ser usados para criar interatividade:

Exemplo 1: Alterar a cor de fundo de um elemento ao passar o mouse sobre ele

```html
<!DOCTYPE html>
<html>
<head>
    <title>Mouseover e Mouseout</title>
    <style>
        #meuDiv {
            width: 200px;
            height: 200px;
            background-color: lightblue;
        }
    </style>
</head>
<body>
    <div id="meuDiv">Passe o mouse aqui</div>

    <script>
        let div = document.getElementById('meuDiv');

        div.addEventListener('mouseover', function() {
            div.style.backgroundColor = 'lightgreen';
        });

        div.addEventListener('mouseout', function() {
            div.style.backgroundColor = 'lightblue';
        });
    </script>
</body>
</html>
```

Exemplo 2: Mostrar uma mensagem ao enviar um formulário

```html
<!DOCTYPE html>
<html>
<head>
    <title>Submit Form</title>
</head>
<body>
    <form id="meuFormulario">
        <input type="text" placeholder="Seu nome">
        <button type="submit">Enviar</button>
    </form>

    <script>
        let formulario = document.getElementById('meuFormulario');

        formulario.addEventListener('submit', function(event) {
            event.preventDefault();
            alert('Formulário enviado!');
        });
    </script>
</body>
</html>
```

ANOTAÇÕES:

QUER APRENDER MAIS?

Visite nosso site e descubra outros cursos de tecnologia que oferecemos!
Prepare-se para aprender e se surpreender.
Mal podemos esperar para ver você brilhando em novas habilidades! Acesse já!

CLIQUE AQUI