Roteiro de Ensino: Construindo uma Calculadora com HTML, CSS e JavaScript

Etapa 1: Estrutura da Calculadora com HTML

Nesta etapa, o foco é criar a estrutura básica da calculadora. O HTML define os elementos da página (display, botões) e suas hierarquias.

- **1.1. O Esqueleto da Página: Explique a estrutura básica de um documento HTML, incluindo as tags <html>, <head> e <body>. Destaque que a tag <head> serve para informações da página (como o título e a ligação com o CSS), enquanto a tag <body> contém todo o conteúdo visível.**
- **1.2. O Contêiner Principal e o Display: Mostre como a div com a classe calculadora age como o contêiner principal do nosso projeto. Dentro dela, o <input> é o display, onde os números e resultados serão exibidos. Explique o atributo id="resultado", que será usado pelo JavaScript para se comunicar com esse elemento, e o atributo readonly, que impede que o usuário digite diretamente no campo.**
- **1.3. Os Botões: Explique que a div com a classe botoes organiza todos os botões. Destaque que cada botão tem a classe botao para estilização. O valor do botão é o texto entre as tags, como 7, + ou C.**

HTML

```
<!DOCTYPE html>

<html>

<head>

 <title>Calculadora</title>

 <link rel="stylesheet" href="style.css">

</head>

<body>

 <div class="calculadora">

  <input type="text" id="resultado" readonly>

  <div class="botoes">

   <button class="botao">7</button>

   <button class="botao">8</button>

   <button class="botao">9</button>

   <button class="botao">/</button>

   <button class="botao">0</button>

   <button class="botao">.</button>

   <button class="botao">+</button>
```

```
    <button class="botao">=</button>
    <button class="botao">C</button>
   </div>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

Etapa 2: Estilizando a Calculadora com CSS

Nesta etapa, a calculadora ganhará sua aparência final. O CSS organizará os botões, definirá cores, tamanhos e adicionará efeitos de transição para uma melhor experiência visual.

- **2.1. O Layout Principal: Explique as regras CSS para a classe calculadora. Propriedades como width, margin e box-shadow dão à calculadora seu tamanho, centralizam-na na página e adicionam uma sombra sutil.**
- **2.2. Estilizando o Display e os Botões: Mostre as regras para o #resultado, explicando como o font-size, padding e text-align fazem com que o display se pareça com o de uma calculadora real. Para os botões, explique o uso de display: grid e grid-template-columns: repeat(4, 1fr) no contêiner .botoes, que cria um layout de grade com quatro colunas de tamanho igual.**
- **2.3. Interatividade Visual: Explique o seletor :hover. Mostre como a regra .botao:hover muda a cor de fundo do botão quando o mouse passa por cima, dando um feedback visual ao usuário.**

CSS

```
.calculadora {
  width: 250px;
  margin: 50px auto;
  border: 1px solid #ccc;
  border-radius: 5px;
  padding: 10px;
  background-color: #f0f0f0;
  box-shadow: 0px 0px 10px rgba(0,0,0,0.2);
}


#resultado {
```

```
  width: 100%;
  height: 40px;
  margin-bottom: 10px;
  padding: 5px;
  font-size: 20px;
  text-align: right;
  border: 1px solid #ccc;
  border-radius: 3px;
  box-sizing: border-box;
}

.botoes {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-gap: 5px;
}

.botao {
  height: 50px;
  font-size: 18px;
  border: none;
  border-radius: 3px;
  background-color: #eee;
  cursor: pointer;
  transition: background-color 0.2s ease;
}

.botao:hover {
  background-color: #ddd;
}
```

Etapa 3: Funcionalidade com JavaScript

Esta é a etapa mais importante, onde a calculadora ganha sua lógica. O JavaScript fará a interação com o HTML, gerenciará o estado das operações e exibirá os resultados.

- **3.1. Gerenciando o Estado da Calculadora: Explique que o JavaScript precisa de variáveis para "lembrar" dos números e do operador. Apresente as variáveis globais (displayValue, firstOperand, etc.) e explique o que cada uma armazena.**
- **3.2. Manipulação do Display e Eventos: Mostre a linha const resultado = document.getElementById('resultado'); para obter o display. Em seguida, explique a função updateDisplay() que atualiza o valor do input. A linha botoes.addEventListener('click', handleButtonClick); é o ponto de partida da interatividade, "ouvindo" os cliques em todos os botões.**
- **3.3. Lógica dos Botões (Números e Operadores):**
    - **Função handleButtonClick: Descreva esta função como a "central de comando" que decide o que fazer com base no botão clicado.**
    - **Função inputDigit: Explique como esta função adiciona números ao display. Destaque a lógica que impede a adição de zeros extras no início.**
    - **Função handleOperator: Esta é a lógica das operações. Explique como ela armazena o primeiro número e o operador, e como ela é chamada novamente para realizar o cálculo quando um novo operador ou o botão = é pressionado.**
- **3.4. O Botão C e o Botão =. Mostre a função resetCalculator(), que limpa todas as variáveis e reinicia a calculadora. Explique que o botão = chama a lógica do handleOperator para finalizar o cálculo e exibir o resultado final.**

JavaScript

```
// 3.1. Gerenciando o estado da calculadora

let displayValue = '';

let firstOperand = null;

let operator = null;

let waitingForSecondOperand = false;


// Objeto para mapear os operadores

const operators = {

  '+': (a, b) => a + b,

  '-': (a, b) => a - b,
```

```
  '*': (a, b) => a * b,
  '/': (a, b) => a / b,
};

// 3.2. Referência ao display e atualização
const resultado = document.getElementById('resultado');
function updateDisplay() {
  resultado.value = displayValue;
}

// 3.3. Lida com a entrada de dígitos
function inputDigit(digit) {
  if (waitingForSecondOperand === true) {
    displayValue = digit;
    waitingForSecondOperand = false;
  } else {
    displayValue = displayValue === '0' ? digit : displayValue + digit;
  }
}

// 3.3. Lida com a entrada de operadores
function handleOperator(nextOperator) {
  const inputValue = parseFloat(displayValue);
  if (operator && waitingForSecondOperand) {
    operator = nextOperator;
    return;
  }
  if (firstOperand === null) {
    firstOperand = inputValue;
  } else if (operator) {
    const result = operators[operator](firstOperand, inputValue);
```

```
    displayValue = `${parseFloat(result.toFixed(7))}`;
    firstOperand = result;
  }
  waitingForSecondOperand = true;
  operator = nextOperator;
}

// 3.4. Lida com o botão 'C'
function resetCalculator() {
  displayValue = '0';
  firstOperand = null;
  operator = null;
  waitingForSecondOperand = false;
  updateDisplay();
}

// 3.2. Lida com cliques nos botões
function handleButtonClick(event) {
  const { target } = event;
  if (!target.matches('button')) {
    return;
  }
  const value = target.textContent;
  if (value === 'C') {
    resetCalculator();
    return;
  }
  if (value === '=') {
    if (operator && firstOperand !== null) {
      handleOperator(operator);
      operator = null;
```

```
    }
    updateDisplay();
    return;
  }
  if (value in operators) {
    handleOperator(value);
  } else {
    inputDigit(value);
  }
  updateDisplay();
}

// 3.2. Adiciona um "ouvinte" de eventos para os botões
const botoes = document.querySelector('.botoes');
botoes.addEventListener('click', handleButtonClick);

// 3.2. Inicializa o display ao carregar a página
document.addEventListener('DOMContentLoaded', () => {
  displayValue = '0';
  updateDisplay();
});
```

@douglasabnovato

24/07/2025