HTML

Esse formulário contém três partes principais: um campo para o usuário digitar o valor que deseja converter, um menu suspenso para escolher a moeda de origem (de onde o dinheiro está saindo) e outro menu suspenso para selecionar a moeda de destino (para onde o dinheiro será convertido). As opções de moedas disponíveis são o Real Brasileiro (BRL), o Dólar Americano (USD) e o Euro (EUR).

Depois desses campos, há um botão para o usuário clicar e iniciar a conversão. O botão tem o tipo “button”, ou seja, ele não envia o formulário de forma tradicional, mas é manipulado por um código JavaScript para realizar a conversão sem recarregar a página.

Logo abaixo do formulário, existem dois espaços reservados para mostrar informações importantes ao usuário: um para o resultado da conversão e outro para exibir mensagens de erro caso algo dê errado, como um valor inválido ou uma escolha incorreta de moeda.

Por fim, o rodapé da página traz uma pequena mensagem de copyright. Além disso, na parte final do corpo, o arquivo “script.js” é referenciado. Esse arquivo será responsável por trazer a funcionalidade dinâmica da página, ou seja, pegar o valor digitado, fazer o cálculo da conversão (provavelmente consultando taxas de câmbio), e mostrar o resultado ou os erros na interface para o usuário.

JS

Esse script foi feito para ser usado junto com o formulário de conversor de moedas. A ideia principal dele é adicionar a funcionalidade, ou seja, fazer o botão “Converter” realmente transformar o valor digitado de uma moeda para outra com base em taxas de câmbio fixas.

No começo, o script espera o carregamento completo do conteúdo da página (DOM) antes de executar qualquer coisa. Isso é importante porque assim ele garante que todos os elementos da página estejam disponíveis para serem usados no código, evitando erros.

Depois, o script seleciona vários elementos da página pelo ID: o campo onde o usuário digita o valor, os dois menus suspensos de moeda (origem e destino), o espaço onde o resultado vai aparecer, o espaço para mensagens de erro, e o botão de converter.

Em seguida, tem um objeto chamado `taxas` que guarda valores fixos para conversão entre os pares de moedas que você colocou no HTML. Por exemplo, de Real para Dólar, a taxa é 0.19, e para o caminho inverso, Dólar para Real, é 5.26. Essas taxas são usadas para calcular o valor convertido. É uma abordagem simples, já que essas taxas não mudam — não tem consulta online.

A parte mais importante vem na função que é chamada quando o botão “Converter” é clicado. O código primeiro pega o valor digitado e tenta transformá-lo em um número decimal. Isso é essencial para garantir que o valor pode ser usado em cálculos.

Depois, ele pega quais moedas foram escolhidas nos dois selects, para saber de onde e para onde converter.

Antes de fazer a conversão, o código limpa qualquer texto que já estivesse aparecendo nos campos de resultado ou erro, deixando a tela pronta para uma nova mensagem.

Logo depois, ele verifica se o valor digitado é válido — ou seja, se é mesmo um número e se é maior que zero. Se não for, mostra uma mensagem dizendo para digitar um valor válido e para por aí.

Outra verificação importante é se a moeda de origem é a mesma que a de destino. Isso não faz sentido converter uma moeda para ela mesma, então o script mostra uma mensagem para o usuário escolher moedas diferentes e também interrompe o processo.

Se tudo estiver certo até aqui, ele monta uma chave, tipo “BRL-USD” ou “USD-EUR”, que serve para buscar a taxa certa dentro do objeto de taxas.

Se ele encontrar essa taxa, calcula o valor convertido multiplicando o valor digitado pela taxa e arredonda o resultado para duas casas decimais, deixando a apresentação mais limpa.

Por fim, mostra esse resultado na tela para o usuário, do jeito “100 BRL = 19.00 USD”, por exemplo.

Se, por acaso, não tiver uma taxa cadastrada para o par escolhido, o código avisa que a conversão não está disponível para aquelas moedas.

CSS

Primeiro, o estilo do **body** já define a base da página. Ele escolhe a fonte Arial (ou uma sans-serif qualquer, caso Arial não esteja disponível), o fundo é um cinza bem clarinho (#f2f2f2), com texto em um tom escuro (#333) para facilitar a leitura. A margem e o padding são zerados para garantir que não tenha espaços indesejados nas bordas da página. O body também vira um contêiner flexível em coluna, que organiza seu conteúdo de cima para baixo e centraliza horizontalmente, garantindo que o conteúdo fique alinhado no meio da tela, tanto em desktops quanto dispositivos móveis. Além disso, a altura mínima é definida para ocupar toda a tela vertical (100vh), ajudando a manter o rodapé sempre na parte inferior.

O **header** ganha um fundo azul vibrante (#007bff) com texto branco, deixando o título do site bem destacado no topo. O padding vertical dá um espaçamento confortável, e o texto fica centralizado, ocupando 100% da largura da tela para criar uma barra completa.

O **main**, que é a área principal onde o formulário fica, tem fundo branco para destacar sobre o fundo cinza do body. Ele tem margens que criam espaço acima, abaixo e nas laterais, além de padding interno para dar espaço entre a borda e os elementos internos. Os cantos são arredondados e há uma sombra suave para dar uma sensação de profundidade e leveza, fazendo com que o conteúdo pareça “flutuar” um pouco sobre o fundo. Ele também

é limitado a uma largura máxima de 400 pixels, para que o conteúdo não fique muito esticado em telas grandes, garantindo melhor leitura.

Os grupos do formulário (.form-group) têm um espaço embaixo, separando cada campo para uma visualização mais limpa.

Os **labels** estão configurados para ocupar toda a linha (display: block), com um espaço abaixo para separar do campo em si, e o texto está em negrito para facilitar a identificação do que deve ser preenchido.

Os campos de input e select têm largura total, garantindo que ocupem todo o espaço disponível dentro do form-group, o que é ótimo para usabilidade, principalmente em dispositivos móveis. Eles têm um padding confortável, bordas arredondadas e uma borda cinza clara para dar uma definição sem ficar pesada. O box-sizing garante que o padding e a borda sejam incluídos na largura total, evitando que o campo estoure fora do contêiner.

O **resultado** da conversão aparece em uma caixa verde clara, com cantos arredondados, texto centralizado e em negrito, usando uma cor verde escura para reforçar a ideia de sucesso ou informação positiva.

Já a **mensagem de erro** aparece em uma caixa vermelha clara, também com cantos arredondados, texto centralizado e cor vermelha escura, sinalizando claramente que algo deu errado, chamando a atenção do usuário para corrigir.

O **footer** fica centralizado, com padding para não ficar “colado” na borda da tela, texto em cinza escuro para não competir com o conteúdo principal, e a margem “auto” no topo faz com que ele seja empurrado para baixo quando o conteúdo for menor que a altura da tela, garantindo que ele fique sempre na parte inferior da página.

Por fim, tem um bloco de estilos **responsivos** para telas pequenas (até 480px, geralmente celulares). Nesses dispositivos, o main recebe margens e padding menores para economizar espaço na tela pequena, e o botão de converter tem o tamanho da fonte reduzido para encaixar melhor sem perder a legibilidade.