Javascript Magic - Parte 2

Lógica do projeto

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Descrição do projeto

Essa é a parte dois do projeto Jogo da memória com Javascript, onde veremos toda a lógica para criar o jogo da memória!

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Primeiro passo: Criar as variáveis globais

Primeiro, precisamos criar nossas variáveis globais, aquelas que serão necessárias para o funcionamento da lógica.

Precisamos estabelecer que o jogador1 que começa primeiro, precisamos guardar a primeira e a ultima carta, e precisaremos de um contador de cliques.

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Primeiro passo: Criar as variáveis globais

```
// Variáveis globais
    let jogador1 = true; // Começa com o primeiro jogador
    let primeiraCarta = null; // Guarda a primeira carta que o usuário clicou na sua vez
    let ultimaCarta; // Vou precisar armazenar a última carta
    let contadorCliques = 0; // Para sabermos quando a carta foi clicada
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Segundo passo: Escrever os dados

Precisamos pegar os nomes dos jogadores que foram digitados na tela inicial, e digitar nessa nova tela do jogo!
Precisamos pegar os dados do localStorage e jogar na tela.

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Segundo passo: Escrever os dados

```
// Função para escrever os nomes dos jogadores
  function escreverDados() {
      //pegando os nomes dos jogadores
      let primeiro_jogador = localStorage.getItem('jogador1');
      let segundo_jogador = localStorage.getItem('jogador2');
      //escrevendo os nomes na tela
      document.getElementById('nome_jogador1').innerHTML = primeiro_jogador;
      document.getElementById('nome_jogador2').innerHTML = segundo_jogador;
  }
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Terceiro passo: Embaralhar as cartas

Precisamos conseguir embaralhar as cartas de forma aleatória a cada novo jogo!
Precisaremos de uma função que faça isso:

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Terceiro passo: Embaralhar as cartas

```
 // Função para embaralhar os cartões
    function shuffle(lista) {
        return lista.sort(() => Math.random() - 0.5);
    }
/*esse código classifica o array lista de forma aleatória. Cada vez que
 você executar esse código, os elementos do array serão reordenados
  aleatoriamente com base nos valores retornados pela função Math.random() - 0.5.*/
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quarto passo: Mostrar os cards na tela

Precisamos de uma função que mostre os cards na tela, mas precisa ter os pares, para ser de fato, um jogo da memória, por isso, a função precisa:
conter a lista de cards
duplicar essa lista
embaralhar a lista
mostrar os cards na tela

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quarto passo: Mostrar os cards na tela

```
// Função para mostrar os cartões na tela
    function mostarCards() {
        // Lista de cartões com suas imagens frontal e traseira
        const cards = [
            { front: "./Assets/back.png", back: "./Assets/card1.png" },
            { front: " ./Assets/back.png", back: "./Assets/card2.png" },
            { front: " ./Assets/back.png", back: "./Assets/card4.png" },
            { front: " ./Assets/back.png", back: "./Assets/card5.png" },
            { front: "./Assets/back.png", back: "./Assets/card6.png" },
            { front: "./Assets/back.png", back: "./Assets/card7.png" },
            { front: "./Assets/back.png", back: "./Assets/card8.png" },
            { front: "./Assets/back.png", back: "./Assets/card9.png" },
            { front: " ./Assets/back.png", back: "./Assets/card10.png" }
        ];
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quarto passo: Mostrar os cards na tela

```
// Embaralha os cartões duas vezes
        let cards_embaralhados1 = shuffle(cards);
        let cards_embaralhados2 = shuffle(cards);
        //junta as duas listas de cartões em uma só
        let cards_geral = cards_embaralhados1.concat(cards_embaralhados2);
        //seleciona o elemento onde vai ficar as cartas
        let container = document.querySelector('.container');
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quarto passo: Mostrar os cards na tela

```
// Loop para criar os elementos de cartão e suas imagens
        for (let i = 0; i < cards_geral.length; i++) {
            // Cria um novo grupo após cada conjunto de 6 cartões
            if (i % 6 == 0) {
                //criamos uma div
                let groupDiv = document.createElement('div');
                //adicionamos a ela a classe group
                groupDiv.classList.add('group');
                //jogamos essa nova div dentro do container
                container.appendChild(groupDiv);
            }
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quarto passo: Mostrar os cards na tela

```
// Cria os elementos de cartão, frente e verso
        let div_card = document.createElement('div');
        let div_back = document.createElement('div');
        let div_front = document.createElement('div');
        let imageBack = document.createElement('img');
        let imageFront = document.createElement('img');
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quarto passo: Mostrar os cards na tela

```
// Adiciona classes aos elementos
        div_card.classList.add('card');
        div_back.classList.add('back');
        div_front.classList.add('front');
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quarto passo: Mostrar os cards na tela

```
// Define as imagens dos cartões
imageBack.src = cards_geral[i].back;
imageFront.src = cards_geral[i].front;
//isso esta dentro do loop, logo, a cada iteração
//do loop, i será um valor diferente
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quarto passo: Mostrar os cards na tela

```
// Adiciona as imagens aos elementos de frente e verso
            div_back.appendChild(imageBack);
            div_front.appendChild(imageFront);
            // Adiciona as faces do cartão ao cartão
            div_card.appendChild(div_back);
            div_card.appendChild(div_front);
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quarto passo: Mostrar os cards na tela

```
// Adiciona o cartão ao último grupo de cartões criado
        container.lastChild.appendChild(div_card);

        // Adiciona um ouvinte de eventos para virar o cartão ao clicar
        div_card.addEventListener('click', function() {
            div_card.classList.toggle('flip');
        });
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quinto passo: Construir a lógica por trás do jogo da memória

Precisamos agora criar a real lógica por trás do jogo.

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quinto passo: Construir a lógica por trás do jogo da memória

```
function logica() {
    // Seleciona o container que contém os cartões
    const container = document.querySelector('.container');
    // Seleciona todos os cartões dentro do container
    const cards = container.querySelectorAll('.card');
    // Declara as variáveis para armazenar as cartas selecionadas
    let firstCard, secondCard;
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quinto passo: Construir a lógica por trás do jogo da memória

```
// Objeto para armazenar os pontos de cada jogador
    let playerScore = { player1: 0, player2: 0 };

    // Variável para rastrear o jogador atual
    let currentPlayer = 'player1';
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

```
// Itera sobre cada carta no conjunto de cartões
  cards.forEach((card) => {

      // Adiciona um ouvinte de eventos para o clique em cada carta
      card.addEventListener('click', () => {

          // Verifica se a carta clicada ainda não foi emparelhada
          if (!card.classList.contains('matched')) {

              // Vira a carta clicada
              card.classList.add('flip');

              // Se firstCard ainda não foi selecionado, define-o como a carta clicada e retorna
              if (!firstCard) {
                  firstCard = card;
                  return;
              }

              // Define secondCard como a carta clicada
              secondCard = card;
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quinto passo: Construir a lógica por trás do jogo da memória

```
// Verifica se as imagens das cartas firstCard e secondCard são iguais
    if (firstCard.querySelector('img').src === secondCard.querySelector('img').src) {

    // Se as imagens forem iguais, adiciona a classe matched para indicar que estão emparelhadas
     firstCard.classList.add('matched');
    secondCard.classList.add('matched');

    // Incrementa o contador de pontos do jogador atual
        playerScore[currentPlayer] += 1;

  // Limpa firstCard e secondCard para permitir a seleção de novas cartas
     firstCard = null;
    secondCard = null;
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quinto passo: Construir a lógica por trás do jogo da memória

```
// Verifica se todas as cartas foram emparelhadas
if (Array.from(container.querySelectorAll('.matched')).length === cards.length) {

    // Se sim, exibe um alerta com o jogador vencedor e sua pontuação
    alert(`Player ${currentPlayer} wins with a score of ${playerScore[currentPlayer]}`);
}

// Alterna para o próximo jogador
currentPlayer = (currentPlayer === 'player1') ? 'player2' : 'player1';
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Quinto passo: Construir a lógica por trás do jogo da memória

```
else {
     // Se as imagens não forem iguais, após um intervalo de 1 segundo,
     // vira as cartas de volta
    setTimeout(() => {
     firstCard.classList.remove('flip');
    secondCard.classList.remove('flip');
    firstCard = null;
    secondCard = null;
    }, 1000);
                }
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Sexto e ultimo passo: Chamar as funções

```
escreverDados();
mostarCards();
logica();
```

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

O projeto está disponível no github, assim como as imagens. Basta clicar no ícone do github abaixo e será redirecionado ao repositório!

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK