AMOR A PRIMERA CARTA

Manual y juego desarrollado:
Yazu Bautista
Diseño y maquetación:
Derek Revilla

MANUAL DEL JUEGO: AMOR A PRIMERA CARTA

Introducción
El juego "Amor a Primera Carta" es un proyecto desarrollado con HTML, CSS y JavaScript, donde el objetivo del jugador es encontrar las parejas de cartas iguales.
Al iniciar la partida, todas las cartas se muestran boca abajo. El jugador deberá dar clic en dos cartas a la vez para intentar descubrir un par idéntico de íconos de corazones 💜💙💝💞💛💗❣️🤎.
Si las cartas coinciden, quedarán reveladas; si no, se voltearán nuevamente después de un segundo.
El juego pone a prueba la memoria visual y la concentración del jugador.

Archivos requeridos
El proyecto utiliza tres archivos principales:
HTML (uno.html) → Estructura base del juego. Contiene el tablero, el botón y la conexión con los demás archivos.
CSS (uno.css) → Define los estilos visuales: colores, posiciones, tipografía, tamaños y organización del tablero.
JavaScript (uno.js) → Controla la lógica del juego: mezcla de cartas, detección de pares, reinicio y manejo de clics.
No se necesitan imágenes externas, ya que el juego utiliza emojis para representar las cartas.
El fondo del juego es un color sólido (#2c3e50) que da un ambiente elegante y moderno.

Estructura HTML — uno.html
Este archivo define el esqueleto del juego y enlaza los estilos y la lógica.
🧩 Explicación detallada:

```
<!DOCTYPE html>
<html lang="en">
```

Declara que el documento es HTML5 y está en inglés.

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>AMOR A PRIMERA CARTA</title>
  <link rel="stylesheet" type="text/css" href="uno.css">
</head>
```

- **<meta charset="utf-8"> permite mostrar correctamente caracteres especiales y emojis.**
- **<meta name="viewport"...> garantiza que el juego se vea bien en móviles y tablets.**
- **<link> conecta con el archivo de estilos externos uno.css.**

```
<body>
  <h1> AMOR A PRIMERA CARTA </h1>
  <div class="game-board"></div>
  <button id="reset-button"> Reset Game </button>
  <script src="uno.js"></script>
</body>
</html>
```

<h1> muestra el título principal del juego.
** será el contenedor donde JavaScript insertará las cartas dinámicamente.**
<button id="reset-button"> permite reiniciar la partida.
<script src="uno.js"> enlaza el archivo de lógica del juego.

Diseño Visual — uno.css
Este archivo da la forma, color y orden a todos los elementos del juego.

Estructura general:

```
body {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
    background-color: #2c3e50;
    color: white;
    font-family: Arial, sans-serif;
}
```

display: flex permite centrar todos los elementos en el cuerpo.
flex-direction: column los coloca en columna (título, tablero, botón).
align-items: center y justify-content: center centran el contenido horizontal y verticalmente.
height: 100vh ocupa toda la altura de la ventana.
background-color: #2c3e50 da un tono oscuro elegante.
color: white hace contraste con el fondo.
font-family: Arial mantiene la legibilidad.

Título:

```
h1 {
    margin-bottom: 20px;
}
```

Separa el título del tablero para que no quede pegado.

Tablero de juego:

```css
.game-board {
    display: grid;
    grid-template-columns: repeat(4, 100px);
    grid-gap: 10px;
}
```

Se usa CSS Grid para organizar las cartas en una cuadrícula.
grid-template-columns: repeat(4, 100px) → crea 4 columnas de 100px cada una.
grid-gap: 10px → deja espacio entre las cartas.
Esto forma un tablero de 4x4 (16 cartas).

Cartas:

```css
.card {
    width: 100px;
    height: 100px;
    background-color: #ecf0f1;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 24px;
    cursor: pointer;
    border-radius: 8px;
}
```

Define el tamaño uniforme de cada carta (100x100px).
background-color: #ecf0f1 → color neutro mientras está "oculta".
display: flex; align-items: center; justify-content: center → centra el emoji en la carta.
font-size: 24px → tamaño visible del ícono.
cursor: pointer → cambia el cursor a mano indicando que se puede hacer clic.
border-radius: 8px → bordes redondeados para un estilo más suave.

Carta volteada:

```css
.card.flipped {
    background-color: #3498db;
    color: white;
}
```

Cuando la carta se "voltea", el fondo cambia a azul (#3498db) y el texto (emoji) se vuelve blanco, simulando la acción de descubrir.

Botón de reinicio:

```css
#reset-button {
    margin-top: 20px;
    padding: 10px 20px;
    font-size: 16px;
    cursor: pointer;
    border: none;
    border-radius: 5px;
    background-color: #e74c3c;
    color: white;
}
```

Da un diseño moderno al botón.
background-color: #e74c3c → rojo intenso que destaca sobre el fondo.
cursor: pointer indica interactividad.
border-radius y padding suavizan su aspecto.

Da un diseño moderno al botón.
background-color: #e74c3c → rojo intenso que destaca sobre el fondo.
cursor: pointer indica interactividad.
border-radius y padding suavizan su aspecto.

Lógica del Juego — uno.js
Aquí se controla toda la interactividad, como los clics, los pares, el reinicio y el estado de las cartas.

Declaración de variables:

```
const cards = [
  '💜', '💜', '💙', '💙', '💝', '💝',
  '💞', '💞', '💛', '💛', '💖', '💖',
  '❣️', '❣️', '🤎', '🤎',
];
```

Crea un arreglo con los íconos duplicados, formando 8 pares.

```
let firstCard = null;
let secondCard = null;
let lockBoard = false;
```

Estas variables controlan el estado actual:
firstCard y secondCard: guardan las cartas volteadas.
lockBoard: evita que el jugador haga clic mientras las cartas se están volteando de nuevo.

Función para mezclar cartas:

```
function shuffle(array) {
  for (let i = array.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1));
    [array[i], array[j]] = [array[j], array[i]];
  }
}
```

Uso del algoritmo Fisher-Yates, que mezcla los elementos del arreglo de manera aleatoria, asegurando partidas distintas cada vez.

Crear el tablero:

```
function createBoard() {
  const gameBoard = document.querySelector('.game-board');
  shuffle(cards);
  cards.forEach(card => {
    const cardElement = document.createElement('div');
    cardElement.classList.add("card");
    cardElement.dataset.icon = card;
    cardElement.textContent = '';
    cardElement.addEventListener('click', flipCard);
    gameBoard.appendChild(cardElement);
  });
}
```

Genera todas las cartas visualmente:
Crea un por cada carta.
Usa dataset.icon para guardar su valor sin mostrarlo.
Agrega el evento click que permite girar la carta.

Genera todas las cartas visualmente:
Crea un por cada carta.
Usa dataset.icon para guardar su valor sin mostrarlo.
Agrega el evento click que permite girar la carta.

Girar carta:

```
function flipCard() {
  if (lockBoard) return;
  if (this === firstCard) return;

  this.classList.add('flipped');
  this.textContent = this.dataset.icon;

  if (!firstCard) {
    firstCard = this;
    return;
  }

  secondCard = this;
  checkForMatch();
}
```

Controla el volteo:
Si el tablero está bloqueado (lockBoard), no se puede jugar.
La carta se "gira" mostrando su emoji (dataset.icon).
Cuando hay dos cartas volteadas, llama a checkForMatch().

Comprobación de pareja:

```
function checkForMatch() {
  if (firstCard.dataset.icon === secondCard.dataset.icon) {
    disableCards();
  } else {
    unflipCards();
  }
}
```

Verifica si las dos cartas son iguales:
Si coinciden → se desactivan.
Si no → se voltean nuevamente.

Desactivar cartas coincidentes:

```
function disableCards() {
  firstCard.removeEventListener('click', flipCard);
  secondCard.removeEventListener('click', flipCard);
  resetBoard();
}
```

Las cartas correctas ya no pueden volverse a hacer clic, simulando que quedan "descubiertas".

Voltear cartas erróneas:

```
function unflipCards() {
  lockBoard = true;
  setTimeout(() => {
    firstCard.classList.remove('flipped');
    secondCard.classList.remove('flipped');
    firstCard.textContent = '';
    secondCard.textContent = '';
    resetBoard();
  }, 1000);
}
```

Usa setTimeout para dejar visibles las cartas erróneas 1 segundo antes de ocultarlas de nuevo.
Durante ese tiempo, lockBoard = true bloquea nuevos clics.

Reinicio del estado:

```
function resetBoard() {
  [firstCard, secondCard, lockBoard] = [null, null, false];
}
```

Limpia las variables para permitir seguir jugando correctamente.

Reiniciar juego completo:

```
document.getElementById('reset-button').addEventListener('click', () => {
  document.querySelector('.game-board').innerHTML = '';
  createBoard();
});
createBoard();
```

Al presionar el botón, el tablero se vacía y se vuelve a crear con las cartas mezcladas nuevamente.

Conclusión

El juego “Amor a Primera Carta” combina simplicidad visual con una lógica clara y eficiente.
Su diseño flexible permite adaptarse a cualquier pantalla y es ideal para proyectos educativos o demostrativos de programación.
Cada valor de estilo y cada función tienen una razón precisa que garantiza que el juego sea intuitivo, funcional y estéticamente agradable.