manual del juego "piedra, papel o tijeras"

manual y juego elaborado por: evelyn paez

diseño y maquetacion: mauricio rojas

traducido por: fernando lopez

introducción

este proyecto presenta el este proyecto presenta el clásico juego "piedra, papel o tijeras", creado con html, css y javascript.
el usuario selecciona una opción (piedra, papel o tijeras) y el programa genera una elección aleatoria para la computadora.
el sistema compara ambas opciones y muestra quién gana, junto con las imágenes de los ataques.
este ejercicio tiene como propósito entender cómo se integran la estructura del sitio (html), los estilos visuales (css) y la lógica del juego (javascript) en una aplicación web interactiva.

documentos utilizados

para crear este proyecto se ocupan los siguientes archivos y recursos:

- index.html → archivo principal que contiene la estructura del juego.
- main.css → archivo que da estilo al contenido (colores, bordes, tamaños y animaciones).
- app.js → archivo que maneja la lógica del juego (funciones, eventos y resultados).
- piedra.png → imagen que representa la opción piedra.
- papel.png → imagen que representa la opción papel.
- tijeras.png → imagen que representa la opción tijeras.

nota: todos los archivos deben estar en la misma carpeta para que el juego funcione correctamente.

imagenes utilizadas

papel

papel.png

tijeras

tigeras.png

piedra

pierda.png

código html [estructura del juego]

el archivo html define toda la estructura visual y los elementos que el usuario verá en la pantalla.

estructura básica del documento

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="main.css">
    <title>Piedra | Papel | Tijeras</title>
</head>
<body>
```

- <!doctype html>: indica que el documento usa html5.
- <html lang="es">: define el idioma español.
- <meta charset="utf-8">: permite usar caracteres especiales como tildes y ñ.
- <link rel="stylesheet" href="main.css">: conecta el archivo css.
- <title>: define el nombre que aparecerá en la pestaña del navegador.

área del campo de batalla

```
<h1>Piedra | Papel | Tijeras</h1>

<div id="campo-batalla" class="campo-batalla">
    <div id="img-ataque-jugador" class="caja-batalla"></div>
    <div class="caja-batalla"><p class="text-mensajes">VS</p></div>
    <div id="img-ataque-pc" class="caja-batalla"></div>
</div>

<div>
    <p id="msj-batalla" class="text-mensajes"></p>
</div>
```

- <h1>: titulo principal del juego.
- : espacio donde se mostrarán las imágenes de las jugadas (jugador vs pc).
- <p id="msj-batalla">: muestra el resultado de la partida ("ganaste", "perdiste" o "empate").

botones de selección (imágenes clicables)

```
<h2>Elije un ataque</h2>
<div>
    <img id="btn-piedra" class="btn-ataques" src="Piedra.PNG" alt="Piedra">
    <img id="btn-papel" class="btn-ataques" src="Papel.PNG" alt="Papel">
    <img id="btn-tijeras" class="btn-ataques" src="Tijeras.PNG" alt="Tijeras">
</div>
```

- cada imagen (<img>) representa un botón para elegir una opción.
- se usan los atributos id para identificarlas desde javascript.
- src indica la ruta de la imagen, y alt describe su contenido.

conexión con javascript

```
<script src="app.js"></script>
</body>
</html>
```

- <script src="app.js">: conecta el archivo javascript, que contiene toda la lógica del juego.

código css (diseño y estilo visual)

el archivo css define la presentación del juego, incluyendo el tamaño de los elementos, colores, márgenes y animaciones.

ajustes generales

```
* {
  box-sizing: border-box;
}

body {
  font-family: 'Josefin Sans', sans-serif;
  display: flex;
  flex-direction: column;
  align-items: center;
  margin: 0;
  padding: 20px;
  background: #fff;
}
```

- box-sizing: border-box: hace que los márgenes y bordes no alteren el tamaño total de los elementos.

body: centra todo el contenido en la página y define la fuente.

campo de batalla

```css
.campo-batalla {
  display: flex;
  justify-content: center;
  align-items: center;
  border: 2px solid red;
  padding: 30px;
  border-radius: 20px;
  gap: 30px;
}
```

- crea un recuadro rojo donde se mostrarán las imágenes de la batalla.
- gap: agrega espacio entre las imágenes.
- border-radius: redondea las esquinas.

---

imágenes y textos

```css
.caja-batalla {
  margin: 25px;
  width: 160px;
  height: 160px;
  display: flex;
  justify-content: center;
  align-items: center;
}

.text-mensajes {
  font-size: 28px;
  font-weight: bold;
  text-align: center;
}

.img-batalla {
  width: 120px;
  height: 120px;
  border-radius: 50%;
  object-fit: cover;
  transition: transform 0.18s ease;
}
```

- .caja-batalla: define el tamaño de cada recuadro del jugador y la computadora.
- .text-mensajes: ajusta el texto del resultado.
- .img-batalla: aplica formato redondo y suaviza la animación de las imágenes.

botones de ataque

```css
.btn-ataques {
  width: 140px;
  height: 140px;
  border-radius: 50%;
  margin: 20px;
  cursor: pointer;
  object-fit: cover;
  transition: transform 0.18s ease, box-shadow 0.18s ease;
  border: none;
  background: transparent;
}

.btn-ataques:hover {
  transform: scale(1.06);
  box-shadow: 0 6px 18px rgba(0, 0, 0, 0.12);
  outline: 2px solid rgba(255, 0, 0, 0.15);
}
```

- .btn-ataques: da forma redonda a las imágenes y las hace clicables.
- :hover: crea un efecto de agrandamiento y sombra al pasar el cursor.
-

código javascript (lógica del juego)

el archivo javascript maneja la interactividad, las elecciones del jugador y de la computadora, y los resultados.

captura de elementos del html

```
const seccionBatalla = document.getElementById('campo-batalla');
const msjBatalla = document.getElementById('msj-batalla');
const imgAtaqueJugador = document.getElementById('img-ataque-jugador');
const imgAtaquePc = document.getElementById('img-ataque-pc');

const btnPiedra = document.getElementById('btn-piedra');
const btnPapel = document.getElementById('btn-papel');
const btnTijeras = document.getElementById('btn-tijeras');
```

se obtienen los elementos del html para manipularlos más adelante [texto, imágenes y botones].

variables e imágenes

```
let opcionJugador;
let opcionPc;

const imagenes = [
    { name: "Piedra", url: "Piedra.PNG" },
    { name: "Papel", url: "Papel.PNG" },
    { name: "Tijeras", url: "Tijeras.PNG" }
];
```

- opcionjugador y opcionpc: guardan las elecciones de cada uno.

imagenes: arreglo con el nombre y la dirección de cada imagen.

inicio del juego

```
window.addEventListener('load', iniciar);

function iniciar() {
    seccionBatalla.style.display = 'none';
}
```

cuando se carga la página, el área de batalla está oculta hasta que el jugador haga una elección.

eventos de clic

```
btnPiedra.addEventListener('click', () => {
    opcionJugador = "Piedra";
    opPc();
});
btnPapel.addEventListener('click', () => {
    opcionJugador = "Papel";
    opPc();
});
btnTijeras.addEventListener('click', () => {
    opcionJugador = "Tijeras";
    opPc();
});
```

cada botón tiene un evento que guarda la elección del jugador y ejecuta la función para generar la elección de la computadora.

elección aleatoria de la computadora

```
function opPc() {
    const aleatorio = nAleatorio();
    opcionPc = imagenes[aleatorio].name;
    batalla();
}

function nAleatorio() {
    return Math.floor(Math.random() * 3);
}
```

- naleatorio(): genera un número entre 0 y 2.
oppc(): usa ese número para elegir entre piedra, papel o tijeras.

batalla y resultado

```
function batalla() {
    if (opcionJugador === opcionPc) {
        msjBatalla.textContent = "Empate 😐";
    } else if (
        (opcionJugador === "Piedra" && opcionPc === "Tijeras") ||
        (opcionJugador === "Papel" && opcionPc === "Piedra") ||
        (opcionJugador === "Tijeras" && opcionPc === "Papel")
    ) {
        msjBatalla.textContent = "¡Ganaste! 🎉";
    } else {
        msjBatalla.textContent = "Perdiste 🤔";
    }
    addImagenes();
}
```

compara las dos elecciones y muestra el resultado según las reglas del juego.

mostrar imágenes en pantalla

```
function addImagenes() {
    const imgJugador = imagenes.find(img => img.name === opcionJugador).url;
    const imgPc = imagenes.find(img => img.name === opcionPc).url;

    imgAtaqueJugador.innerHTML = `<img class="img-batalla" src="${imgJugador}" alt="${opc
    imgAtaquePc.innerHTML = `<img class="img-batalla" src="${imgPc}" alt="${opcionPc}">`;

    seccionBatalla.style.display = 'flex';
}
```

inserta las imágenes correspondientes de las elecciones del jugador y la computadora dentro del campo de batalla.

resultado final
al ejecutar el código:
se muestran las imágenes de piedra, papel y tijeras.

1. el jugador selecciona una opción.
2. la computadora elige aleatoriamente otra.
3. aparecen ambas imágenes y el resultado ("ganaste", "perdiste" o "empate").