Código HTML

Estructura Básica del documento HTML

<!DOCTYPE html> : Le dice al navegador que el documento está escrito en HTML5, la última versión de HTML

<html lang="es">: Indica que el lenguaje del contenido es español (es).

Cabecera del documento (<head>)

<head>

<meta charset="UTF-8">: Especifica el conjunto de caracteres que utilizará el documento, en este caso UTF-8, que soporta la mayoría de los caracteres (incluyendo acentos y símbolos).

<title>Aplicación de Pokemones</title>: El título que se muestra en la pestaña del navegador cuando se abre la página.

<link rel="stylesheet" href="styles.css">: Con esto, el archivo de estilos CSS (styles.css) se vincula a la página, permitiendo que los estilos que definas en ese archivo se apliquen al HTML (como colores, tamaños, posicionamiento de elementos, etc.).

<meta name="viewport" content="width=device-width, initial-scale=1.0">: Hace que la página sea **responsive**, lo que significa que se ajusta automáticamente a diferentes tamaños de pantalla (como dispositivos móviles y computadoras). Esta etiqueta asegura que el ancho de la página se ajuste al ancho del dispositivo en el que se visualiza.

</head>

Cuerpo del documento (<body>)

<body>

<h1>Pokédex</h1>: Esto crea un título de nivel 1 (<h1>), que dice **"Pokédex"**. Es el encabezado principal de la página y se verá en la parte superior en texto grande.

<input type="text" id="search" placeholder="Buscar Pokémon">: Este es un **campo de búsqueda** donde los usuarios pueden escribir el nombre de un Pokémon que desean buscar. El atributo placeholder="Buscar Pokémon" muestra un texto de sugerencia cuando el campo está vacío.

** :** Un **contenedor ( )** donde se mostrarán los Pokémon obtenidos de la API. Inicialmente está vacío, pero el JavaScript (app.js) llenará este div con los datos de los Pokémon.

Modal de detalles del Pokémon

** :** Es el modal que se muestra cuando se selecciona un Pokémon. Está diseñado para mostrar detalles sobre el Pokémon específico.

** :** Es el contenido del modal, que incluye el nombre, imagen, altura, peso y habilidades del Pokémon.

×: Este es el botón de cerrar (con una ×, que es el símbolo de "X") para cerrar el modal.

<h2 id="pokemon-name"></h2> Aquí se mostrará el nombre del Pokémon seleccionado.

<img id="pokemon-img" src="" alt="Imagen del Pokémon">: Este es un espacio reservado para la imagen del Pokémon. La imagen se cargará dinámicamente cuando se selecciona un Pokémon.

** Altura: :** Muestra la **altura** del Pokémon en el campo pokemon-height.

** Peso: :** Muestra el **peso** del Pokémon en el campo pokemon-weight.

** Habilidades: :** Muestra las **habilidades** del Pokémon en el campo pokemon-abilities.

** **

** **

Vinculación del archivo JavaScript

<script src="app.js"></script>: Este archivo JavaScript (app.js) contiene el código que maneja la interacción con la API de PokeAPI y manipula el DOM (Document Object Model) para mostrar los Pokémon y manejar eventos como la búsqueda o la selección de Pokémon.

</body>

</html>

Código CSS

Estilo general del cuerpo (<body>)

body {

font-family: Arial, sans-serif; : Define la fuente que se va a usar en todo el documento. En este caso, Arial es la fuente principal, y sans-serif es la fuente de respaldo en caso de que Arial no esté disponible.

text-align: center; : Centra todo el contenido en la página (texto e imágenes

margin: 0; padding: 0; : Elimina los márgenes y el padding (relleno) por defecto que suelen tener los navegadores para evitar que haya espacios alrededor del contenido.

}

Estilo del titulo (<h1>)

h1 {

background-color: #ef5350; Establece un color de fondo rojo (color hexadecimal #ef5350).

color: white; : Cambia el color del texto a blanco.

padding: 20px; : Añade un espacio interno de 20 píxeles alrededor del contenido del título.

margin: 0; Elimina cualquier margen externo que el navegador pueda añadir por defecto al título.

}

Estilo del campo de búsqueda (<input id="search">)

#search {

width: 80%; El campo de búsqueda ocupará el 80% del ancho disponible de la pantalla.

padding: 10px; Añade 10 píxeles de espacio interno dentro del campo de texto.

margin: 20px auto; Añade un margen de 20 píxeles arriba y abajo, y centra el campo horizontalmente.

font-size: 16px; Establece el tamaño de la fuente dentro del campo de búsqueda a 16 píxeles.

}

Estilo del contenedor para la lista de Pokemones ( )

#pokemon-list {

display: flex; Usa el modelo de caja flexible (Flexbox) para organizar el contenido dentro del contenedor.

flex-wrap: wrap; Permite que los elementos dentro del contenedor (las tarjetas de Pokémon) se ajusten en varias filas si no caben en una sola.

justify-content: center; Centra las tarjetas de Pokémon horizontalmente dentro del contenedor.

}

Estilo de las tarjetas de Pokémon ( )

.pokemon {

border: 1px solid #ddd; Añade un borde de 1 píxel de ancho con un color gris claro (#ddd) alrededor de cada tarjeta.

border-radius: 10px; Redondea las esquinas del borde con un radio de 10 píxeles.

margin: 10px; Añade un margen de 10 píxeles alrededor de cada tarjeta para separarlas.

padding: 10px; Añade 10 píxeles de espacio interno dentro de la tarjeta.

width: 150px; Establece el ancho fijo de la tarjeta en 150 píxeles.

text-align: center; Centra el texto y el contenido dentro de la tarjeta.

}

Estilo de la imagen dentro de la tarjeta de Pokémon

.pokemon img {

width: 100px; La imagen del Pokémon tendrá un ancho de 100 píxeles.

height: 100px; La altura de la imagen también será de 100 píxeles, haciendo que las imágenes sean cuadradas.

}

Estilo de los botones (<button class="button">)

.button { Aplica estos estilos a cualquier botón que tenga la clase button.

background-color: #ef5350; Establece un color de fondo rojo (mismo que el título).

color: white; Cambia el color del texto del botón a blanco.

border: none; Elimina cualquier borde alrededor del botón.

padding: 5px; Añade 5 píxeles de espacio interno alrededor del texto del botón.

margin-top: 5px; Añade un margen de 5 píxeles en la parte superior del botón para separarlo de otros elementos.

cursor: pointer; Cambia el cursor a una mano cuando el usuario pasa el ratón sobre el botón, indicando que es un elemento interactivo.

border-radius: 5px; Redondea las esquinas del botón con un radio de 5 píxeles.

}

.button:hover { Define los estilos cuando el usuario pasa el ratón sobre el botón (hover).

background-color: #d32f2f; Cambia el color de fondo a un rojo más oscuro cuando el botón es seleccionado por el ratón.

}

Estilo del modal (ventana emergente)

.modal {

display: none; El modal está oculto por defecto. Se mostrará solo cuando se active con JavaScript

position: fixed; El modal se posiciona de manera fija en la pantalla, sin importar el scroll.

z-index: 1; Asegura que el modal esté por encima de todos los demás elementos de la página.

left: 0; top: 0; El modal comienza desde la esquina superior izquierda de la página.

width: 100%; height: 100%; El modal ocupará todo el ancho y alto de la pantalla.

overflow: auto; Si el contenido del modal es demasiado grande, permite que aparezcan barras de desplazamiento.

background-color: rgba(0,0,0,0.5); El fondo del modal es una capa semi-transparente de color negro (rgba(0,0,0,0.5)), lo que da un efecto de sombreado sobre el resto de la página

}

Estilo del contenido del modal

.modal-content {

background-color: #fefefe; El fondo del contenido del modal es blanco.

margin: 10% auto; Añade un margen superior del 10% para centrar el contenido verticalmente, y auto para centrarlo horizontalmente.

padding: 20px; Añade 20 píxeles de espacio interno alrededor del contenido del modal.

border: 1px solid #888; Agrega un borde de 1 píxel de ancho con un color gris claro.

width: 80%; El modal ocupará el 80% del ancho de la pantalla.

max-width: 500px; Establece un ancho máximo de 500 píxeles, para que no sea demasiado grande en pantallas grandes.

position: relative; Posiciona los elementos de forma relativa dentro del modal.

border-radius: 10px; Redondea las esquinas del contenido del modal con un radio de 10 píxeles.

}

Estilo del botón de cerrar ()

.close {

color: #aaa; Establece el color del texto en un gris claro.

position: absolute; Posiciona el botón de cerrar en la esquina superior derecha del modal.

top: 10px; right: 25px; Establece la posición del botón a 10 píxeles desde arriba y 25 píxeles desde la derecha.

font-size: 28px; Define el tamaño de la fuente a 28 píxeles.

font-weight: bold; Hace que el texto sea negrita (más grueso).

cursor: pointer; Cambia el cursor a una mano cuando el ratón pasa sobre el botón, indicando que se puede hacer clic.

}

.close:hover, .close:focus: Cambia el color del botón de cerrar a negro cuando el ratón pasa sobre él o cuando se selecciona con el teclado (focus).

{

color: black;

}

Código JavaScript

Selección de elementos del DOM

const pokemonList = document.getElementById('pokemon-list'); Selecciona el contenedor donde se mostrará la lista de Pokemones. Este contenedor tiene el ID pokemon-list en el HTML.

const pokemonDetails = document.getElementById('pokemon-details'); Selecciona el modal (ventana emergente) que mostrará los detalles de un Pokémon cuando se haga clic en él. Su ID es pokemon-details.

const closeModal = document.getElementsByClassName('close')[0]; Selecciona el botón de cierre de la ventana modal, que tiene la clase close. Como getElementsByClassName devuelve una colección de elementos, [0] selecciona el primer (y único) elemento con esa clase.

const searchInput = document.getElementById('search'); Selecciona el campo de búsqueda de la página, identificado por su ID search.

Función para obtener la lista de Pokemones

function fetchPokemonList() {

fetch('https://pokeapi.co/api/v2/pokemon?limit=10') Esta función realiza una llamada a la API de Pokémon (https://pokeapi.co/api/v2/pokemon?limit=10) para obtener los datos de 10 Pokemones.

.then(response => response.json()) Convierte la respuesta en un objeto JSON.

.then(data => {

// Por cada Pokémon, obtenemos sus detalles

data.results.forEach(pokemon => {

fetchPokemonDetails(pokemon.url);

});

})

.catch(error => console.error('Error al obtener la lista de Pokemones:', Captura y muestra en la consola cualquier error que ocurra durante la llamada a la API.**error));**

}

// Llamamos a la función al cargar la página

fetchPokemonList();

Función para obtener detalles de un Pokémon

function fetchPokemonDetails(url) { Esta función toma la URL de un Pokémon y hace una solicitud para obtener sus detalles (como nombre, imagen, altura, etc.).

fetch(url)

.then(response => response.json())

.then(pokemon => {

createPokemonCard(pokemon); Una vez que los detalles del Pokémon se obtienen, esta función crea la tarjeta del Pokémon (que es el siguiente paso).

})

.catch(error => console.error('Error al obtener detalles del Pokémon:', error));

}

Función para crear la tarjeta de un Pokémon

function createPokemonCard(pokemon) { Crea una nueva tarjeta para el Pokémon

const pokemonCard = document.createElement('div'); Crea un div para la tarjeta del Pokémon y le añade la clase pokemon.

pokemonCard.classList.add('pokemon');

// Imagen del Pokémon

const pokemonImg = document.createElement('img');

pokemonImg.src = pokemon.sprites.front_default; Establece la imagen del Pokémon.

pokemonImg.alt = pokemon.name;

// Nombre del Pokémon

const pokemonName = document.createElement('p');

pokemonName.textContent = pokemon.name;

- Botón de favorite se crea un botón que permite marcar el Pokémon como favorito.

const favoriteButton = document.createElement('button');

favoriteButton.textContent = 'Favorito';

favoriteButton.classList.add('button');

favoriteButton.addEventListener('click', () => {

toggleFavorite(pokemon.name, favoriteButton);

});

- Crea un botón que abre un modal con los detalles del Pokémon.

const detailsButton = document.createElement('button');

detailsButton.textContent = 'Detalles';

detailsButton.classList.add('button');

detailsButton.addEventListener('click', () => {

showPokemonDetails(pokemon);

});

- Añade todos estos elementos a la tarjeta, y luego agrega la tarjeta a la lista de Pokemones en el DOM.

pokemonCard.appendChild(pokemonImg);

pokemonCard.appendChild(pokemonName);

pokemonCard.appendChild(favoriteButton);

pokemonCard.appendChild(detailsButton);

// Agregamos la tarjeta al contenedor de la lista

pokemonList.appendChild(pokemonCard);

// Verificamos si el Pokémon es favorito

checkIfFavorite(pokemon.name, favoriteButton); Verifica si el Pokémon ya está marcado como favorito (y ajusta el estilo del botón en consecuencia).

}

Función para mostrar los detalles del Pokémon en el modal

function showPokemonDetails(pokemon) { Muestra un modal con los detalles del Pokémon

- Cambia el contenido del modal para mostrar el **nombre**, **imagen**, **altura**, **peso** y **habilidades** del Pokémon seleccionado.

document.getElementById('pokemon-name').textContent = pokemon.name;

document.getElementById('pokemon-img').src = pokemon.sprites.front_default;

document.getElementById('pokemon-height').textContent = pokemon.height;

document.getElementById('pokemon-weight').textContent = pokemon.weight;

// Obtenemos las habilidades y las unimos en una cadena

const abilities = pokemon.abilities.map(ability => ability.ability.name).join(', ');

document.getElementById('pokemon-abilities').textContent = abilities;

// Mostramos el modal

pokemonDetails.style.display = 'block'; Muestra el modal.

}

Cerrar el modal

closeModal.onclick = function() { Cuando se hace clic en la "X" del modal, este se cierra.

pokemonDetails.style.display = 'none';

}

// Cuando el usuario hace clic fuera del modal, también lo cerramos

window.onclick = function(event) { Si el usuario hace clic fuera del modal, este también se cierra.

if (event.target == pokemonDetails) {

pokemonDetails.style.display = 'none';

}

}

Función para alternar el estado de favorito

function toggleFavorite(pokemonName, button) { Alterna el estado de favorito de un Pokémon.

let favorites = JSON.parse(localStorage.getItem('favorites')) || [];

if (favorites.includes(pokemonName)) {

// Si ya es favorito, lo removemos

favorites = favorites.filter(name => name !== pokemonName);

button.style.backgroundColor = '#ef5350'; // Color por defecto

} else {

// Si no es favorito, lo agregamos

favorites.push(pokemonName);

button.style.backgroundColor = '#FFD700'; // Color dorado para indicar favorite

}

// Guardamos la lista actualizada en localStorage

localStorage.setItem('favorites', JSON.stringify(favorites));

}

Verificar si un Pokémon es favorito al cargar la página

function checkIfFavorite(pokemonName, button) { Comprueba si el Pokémon ya está en la lista de favoritos al cargar la página y cambia el estilo del botón si es necesario.

let favorites = JSON.parse(localStorage.getItem('favorites')) || [];

if (favorites.includes(pokemonName)) {

button.style.backgroundColor = '#FFD700'; // Color dorado para indicar favorito

}}

Filtrar Pokemones mientras el usuario escribe

searchInput.addEventListener('keyup', () => { Escucha el evento keyup en el campo de búsqueda, que se dispara cada vez que el usuario escribe.

const searchTerm = searchInput.value.toLowerCase(); Captura el término que el usuario está escribiendo en minúsculas.

const pokemonCards = document.getElementsByClassName('pokemon');

Array.from(pokemonCards).forEach(card => { Convierte la colección de tarjetas de Pokémon en un arreglo para poder iterar sobre ellas.

const pokemonName = card.getElementsByTagName('p')[0].textContent.toLowerCase();

if (pokemonName.includes(searchTerm)) { Si el nombre del Pokémon incluye el término de búsqueda, la tarjeta se muestra, de lo contrario, se oculta (card.style.display = 'none';).

card.style.display = 'block';

} else {

card.style.display = 'none';

});});