**Promesas en JavaScript**

Las promesas son una forma de manejar operaciones asíncronas como llamadas a servidores o eventos. Representan el estado y resultado de un proceso asíncrono y pueden estar en tres estados:

1. **Pending (pendiente)**: El proceso aún no ha terminado.
2. **Fulfilled (cumplida)**: El proceso se ha completado exitosamente.
3. **Rejected (rechazada)**: Ha ocurrido un error.

Las promesas se crean usando Promise y aceptan una función que recibe dos callbacks:

* **resolve**: Se ejecuta cuando la promesa se cumple.
* **reject**: Se ejecuta si la promesa falla.

Métodos importantes de una promesa:

* .then(): Ejecuta una función cuando la promesa se resuelve con éxito.
* .catch(): Ejecuta una función si la promesa es rechazada.
* .finally(): Ejecuta una función al finalizar la promesa, haya sido resuelta o rechazada.

**Fetch API**

La API fetch es una alternativa moderna a XMLHttpRequest para realizar peticiones HTTP y está basada en promesas. fetch devuelve una promesa que maneja respuestas y errores:

1. El primer .then() recibe un objeto Response, que contiene la respuesta HTTP.
2. .json(), .text(), y otros métodos permiten extraer datos del Response, devolviendo otra promesa.
3. .catch() maneja errores en la conexión.

Propiedades de Response:

* **status** y **statusText**: Código y texto del estado de respuesta.
* **ok**: Es true si la respuesta es un código 20X, false en caso contrario.
* Métodos como .json(), .text(), y .blob() devuelven los datos en diferentes formatos.

Opciones de fetch: Acepta opciones adicionales (método, cabeceras, cuerpo) como objeto de configuración para modificar la petición.

**Manipulación del DOM**

El DOM (Document Object Model) permite acceder y modificar elementos HTML mediante JavaScript. Métodos de acceso a nodos:

* **getElementById()**: Obtiene un nodo por su id.
* **getElementsByClassName()** y **getElementsByTagName()**: Obtienen nodos por clase o etiqueta, devolviendo colecciones.
* **querySelector()** y **querySelectorAll()**: Seleccionan nodos mediante selectores CSS.

Propiedades y métodos importantes de nodos:

* **innerHTML**: Devuelve el contenido HTML de un elemento.
* **textContent**: Devuelve solo el texto.
* **appendChild()**, **insertBefore()**, **removeChild()** y **replaceChild()**: Permiten agregar, insertar, eliminar o reemplazar nodos.

La interfaz **ChildNode** ofrece métodos adicionales para manipular el DOM (before(), after(), replaceWith(), remove()). También se pueden gestionar atributos de nodos con métodos como **getAttribute()**, **setAttribute()**, **removeAttribute()**.

**BOM (Browser Object Model)**

El BOM permite interactuar con el navegador y obtener información de la pantalla, controlar ventanas, manejar cookies y establecer temporizadores.

**Temporizadores**

* **setTimeout()**: Ejecuta una función después de un tiempo en milisegundos.
* **setInterval()**: Ejecuta una función repetidamente a intervalos de tiempo especificados.

**Almacenamiento local y de sesión**

El BOM incluye **localStorage** y **sessionStorage** para almacenar datos del lado del cliente en formato clave-valor:

* **localStorage**: Almacena datos indefinidamente (persisten al cerrar el navegador).
* **sessionStorage**: Almacena datos solo durante la sesión del navegador.

Propiedades y métodos de almacenamiento:

* **setItem()**, **getItem()**, **removeItem()**, y **clear()**: Permiten manipular los datos almacenados.

**Propiedades de location y history**

* **location**: Proporciona información sobre la URL actual, y métodos para recargar o redirigir la página.
* **history**: Permite navegar en el historial de páginas con métodos como .back(), .forward(), y .go().

Este conjunto de funcionalidades, que incluye Promesas, API Fetch, manipulación del DOM y BOM, permite desarrollar aplicaciones interactivas que manejan eficientemente las interacciones con el servidor, manipulan contenidos de la página y permiten interacciones avanzadas con el navegador y el usuario.

let frutas = {};

let pedidoFrutas = {};

let totalEuros = 0;

let totalKg = 0;

// Función para cargar el JSON y almacenarlos en frutaElegida:

function cargarFrutasJson() {

  fetch("../frutas.json") // Inicia la solicitud y devuelve una promesa

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

    .then((frutasData) => {

      //recorre el array de frutas

      for (let i = 0; i < frutasData.frutas.length; i++) {

        let fruta = frutasData.frutas[i]; // obtiene cada fruta

        //almacena la fruta

        frutas[fruta.nombre] = {

          precio: fruta.precioKilo,

          temporada: fruta.temporada,

          proximidad: fruta.proximidad || false,

          region: fruta.region || null,

          nevera: fruta.nevera || false,

          kilos: 0,

        };

      }

      agregarClickFrutas();

    })

    .catch((error) => console.error(error)); // Manejo de errores

}

//Modifico la función de añadirFruta y registrar los kilos de cada fruta seleccionada:

function añadirFruta(nombreFruta, kilos) {

  if (kilos > 0) {

    // Para agregar la fruta si NO está en el pedido

    if (!pedidoFrutas[nombreFruta]) {

      pedidoFrutas[nombreFruta] = {}; // Creo un objeto

      pedidoFrutas[nombreFruta].kilos = 0; // Inicio kg

      pedidoFrutas[nombreFruta].precio = frutas[nombreFruta].precio; // precio

    }

    // para sumar kg de la fruta al pedido

    pedidoFrutas[nombreFruta].kilos += kilos;

    // Actualiza kg y €

    totalKg += kilos; // Suma kg al total de kg

    totalEuros += frutas[nombreFruta].precio \* kilos; // Suma el total al total de euros

    // Actualiza el resumen

    actualizarApartadoResumen(nombreFruta);

    actualizarResumen(nombreFruta);

  } else {

    alert("Introduce un número válido de kg.");

  }

}

function actualizarApartadoResumen(nombreFruta, kilos) {

  const zonaResumenCarro = document.getElementById("resumenCarro");

  // comprueba si ya existe una linea con esta fruta

  const frutaLinea = document.getElementById(`resumen-${nombreFruta}`);

  if (frutaLinea) {

    // Si existe, se actualiza la línea

    frutaLinea.textContent = `${nombreFruta} ---- ${pedidoFrutas[nombreFruta].kilos} kilos`;

  } else {

    //si no existe, creo una nueva linea

    const nuevaLinea = document.createElement("div");

    nuevaLinea.classList.add("fila-fruta");

    nuevaLinea.textContent = `${nombreFruta} --- ${kilos} kilos`;

    // sombreado:

    if (pedidoFrutas[nombreFruta].kilos > 0) {

      nuevaLinea.classList.add("sombreado");

    }

    // Para eliminar el sombreado de cualquier otra línea anterior

    const lineasFrutas = zonaResumenCarro.querySelectorAll(".fila-fruta");

    for (let i = 0; i < lineasFrutas.length; i++) {

      const linea = lineasFrutas[i];

      if (

        linea.classList.contains("sombreado") &&

        !linea.textContent.includes(nombreFruta)

      ) {

        linea.classList.remove("sombreado");

      }

      // Sombrear todas las líneas previas del mismo nombre de fruta

      if (linea.textContent.includes(nombreFruta)) {

        linea.classList.add("sombreado");

      }

    }

    // Añadir la nueva línea en la zona lateral derecha

    zonaResumenCarro.appendChild(nuevaLinea);

  }

}

function finalizarPedido() {

  if (totalKg === 0) {

    alert("No hay nada en la cesta.");

    return;

  }

  let resumenPedido = document.getElementById("resumen");

  resumenPedido.innerHTML = ""; // Limpia el resumen anterior

  // Para mostrar fecha de compra:

  let fecha = new Date();

  let fechaTexto =

    fecha.getDate() +

    "/" +

    (fecha.getMonth() + 1) + //xq comienza x 0. 0=Enero

    "/" +

    fecha.getFullYear() +

    " " +

    fecha.getHours() +

    ":" +

    fecha.getMinutes();

  resumenPedido.innerHTML += "Fecha de compra: " + fechaTexto + "<br>";

  // Para inicializar el total del pedido

  let totalPedido = 0;

  // Bucle que itera sobre pedidoFrutas

  for (const nombre in pedidoFrutas) {

    // 'for...in' para recorrer el objeto

    let kilos = pedidoFrutas[nombre].kilos; // Obtiene kg

    let precio = pedidoFrutas[nombre].precio; // Obtiene €

    let importeTotal = kilos \* precio; // Calcula total de esa fruta

    totalPedido += importeTotal; // Suma al total

    resumenPedido.innerHTML += `${nombre} .... ${kilos} kilos ....

    ${precio.toFixed(2)}€ .... ${importeTotal.toFixed(2)}€ <br>`;

  }

  resumenPedido.innerHTML += `Precio total: ${totalPedido.toFixed(2)} + €<br>`;

  if (totalKg > 0) {

    const precioMedio = totalEuros / totalKg;

    resumenPedido.innerHTML += `Precio medio: ${precioMedio.toFixed(3)} €/kg`;

    // Para hacer desaparecer los pedidos después de 10 segundos:

    setTimeout(quitarPedido, 10000);

  }

}

//Vuelve a 0 la cesta después de 10s:

function quitarPedido() {

  pedidoFrutas = {};

  totalEuros = 0;

  totalKg = 0;

  document.getElementById("resumenCarro").innerHTML = "";

  document.getElementById("resumen").innerHTML = "";

}

//A esta funcion le añado los kilos

function agregarClickFrutas() {

  const frutaImagen = document.querySelectorAll(".fruta img");

  frutaImagen.onclick = function () {

    let kilos = parseInt(frutaImagen.value) || 0;

    añadirFruta(fruta.nombre, kilos);

  };

}

// Para que se inicie la carga de frutas cuando se cargue la página:

window.onload = cargarFrutasJson;

function mostrarResumenCompra() {

  document.getElementById("apartadoResumenCompra").classList.remove("oculto"); // Mostrar resumen de compra

  document.getElementById("apartadoResumen").classList.add("oculto"); // Ocultar carrito

  finalizarPedido();

}

// Asignar la función al botón "Ir a resumen"

document.getElementById("botonFinalizar").onclick = mostrarResumenCompra;