 Vectores JavaScript

Panaderia Todo en Panes

HTML 5

 Edwin Gua

HTML5

CODIGO HTML5?

1. **Configuración y Metadatos (<head>)**
   - **Identidad:** Define el título de la pestaña ("Panadería – Todo En Panes").
   - **Adaptabilidad:** La etiqueta viewport asegura que la página se vea bien en celulares.
   - **Conexiones:** Enlaza el diseño visual (style.css) y el idioma (español).

2. **Encabezado y Navegación (<header>)**
   - Muestra el nombre del negocio y crea un menú con **enlaces internos** (anclas) para saltar rápidamente a las secciones de Inicio, Productos o Caja sin recargar la página.

3. **Contenido Principal (<main>)**

El cuerpo del sitio se divide en cuatro funciones lógicas:

   - **Presentación (#inicio):** Expone la marca mediante el logo, la misión y la visión de la empresa.
   - **Captura de Datos (#cliente):** Proporciona un formulario para recoger el nombre, dirección y **NIT** del comprador. Incluye un validador visual para errores de escritura.
   - **Catálogo Interactivo (#productos):** Es una sección dinámica. Los botones de "Panes", "Pasteles" y "Postres" están listos para filtrar productos mediante funciones que se activarán con JavaScript.
   - **Gestión de Ventas (#carrito):** Es el área de facturación. Muestra lo que el usuario ha seleccionado y contiene el botón para finalizar la compra y descontar del inventario.

4. **Lógica y Dinamismo (<script>)**
   - Al final, llama al archivo js.js. Este es el "cerebro" que hará que el carrito sume, que el NIT se valide y que los productos aparezcan en pantalla cuando hagas clic en los botones.

HTML5

CODIGO CSS?

```css
body { font-family: 'Arial', sans-serif; margin: 0; background-color: #fcf8f3; color: #4e342e; }
.container { padding: 20px; max-width: 1100px; margin: auto; }
header { background: #6d4c41; color: white; padding: 20px; text-align: center; position: sticky; top: 0; z-index: 100; }
nav a { color: #d7ccc8; margin: 0 15px; text-decoration: none; font-weight: bold; }
nav a:hover { color: white; }
```

1. **Estilo General y Base (body y .container)**
   - **Atmósfera:** Usa un color crema de fondo (#fcf8f3) y letras café oscuro (#4e342e) para dar una sensación de "panadería artesanal".
   - **Alineación:** El contenedor centra todo el contenido en la pantalla con un ancho máximo de 1100px para que no se vea desparramado en monitores grandes.

2. **Encabezado Fijo (header y nav)**
   - **Sticky:** La función position: sticky; top: 0; hace que el menú se quede **siempre visible arriba** mientras el usuario baja por la página.
   - **Interactividad:** El efecto hover en los enlaces de navegación hace que cambien de color cuando pasas el ratón, indicando que se puede hacer clic.

HTML5

CSS

```css
.section { background: white; padding: 25px; margin-bottom: 30px; border-radius: 12px; box-shadow: 0 4px 15px rgba(0,0,0,0.05); }
.info-grid { display: flex; flex-wrap: wrap; gap: 20px; align-items: center; }
.hero-img { width: 100%; max-width: 400px; border-radius: 10px; }

.form-group input { padding: 12px; margin: 10px 5px; border: 1px solid #d7ccc8; border-radius: 5px; width: 250px; }

.grid-productos { display: grid; gap: 20px; grid-template-columns: repeat(auto-fit, minmax(220px, 1fr)); margin-top: 20px; }
.card { border: 1px solid #efebe9; padding: 15px; border-radius: 10px; text-align: center; transition: transform 0.2s; }
.card:hover { transform: translateY(-5px); box-shadow: 0 5px 15px rgba(0,0,0,0.1); }
.card img { width: 100%; height: 140px; object-fit: cover; border-radius: 8px; }
```

3. Tarjetas y Secciones (.section, .card)

- **Limpieza Visual:** Las secciones aparecen en cajas blancas con bordes redondeados y una sombra sutil (box-shadow), lo que da profundidad.
- **Animación:** Las tarjetas de productos (.card:hover) tienen un efecto de **elevación** (translateY) para que el usuario sienta que el producto "resalta" al seleccionarlo.

4. Cuadrícula de Productos (.grid-productos)

- Usa **CSS Grid** con auto-fit. Su función es acomodar automáticamente los panes y pasteles: si hay espacio pone 4 por fila, si la pantalla es pequeña pone 2, y así sucesivamente.

HTML 5

CSS

```css
.tabs { margin-bottom: 20px; }
.tabs button { padding: 10px 20px; background: #8d6e63; color: white; border: none; cursor: pointer; border-radius: 5px; margin-right: 10px; }
.btn-comprar { background: #2e7d32; color: white; border: none; padding: 15px 30px; cursor: pointer; border-radius: 5px; width: 100%; font-size: 1.1rem; }

/* Responsividad */
@media (max-width: 600px) {
    .form-group input { width: 100%; margin: 5px 0; }
    .info-grid { flex-direction: column; }
```

5. Formularios y Botones (input, .tabs, .btn-comprar)
   - Estética de entrada: Los campos de texto (Nombre, NIT) tienen bordes suaves y espacio interno para que sea cómodo escribir.
   - Botón de Acción: El botón de "Finalizar Compra" es verde (#2e7d32) y grande, resaltando que es la acción más importante.

6. **Diseño Responsivo (@media)**
   - Su función es la **adaptabilidad móvil**. Cuando la pantalla mide menos de 600px, los campos de texto se ensanchan al 100% y los elementos (como la misión y visión) se apilan uno debajo de otro en lugar de estar de lado a lado.

HTML5

JS

```
// Vectores de información (Vectores solicitados)
const nombresPanes = ["Pan de Manteca", "Pan Integral", "Baguette", "Pan de Yema", "Dobladas"];
const preciosPanes = [1.50, 2.00, 5.00, 2.50, 1.25];
const stockPanes = [100, 150, 130, 160, 180];
const imgPanes = ["pan1.jpg", "pan2.webp", "pan3.webp", "pan4.jpg", "pan5.jpg"];

const inventario = {
    panes: [
        { id: 0, nombre: nombresPanes[0], precio: preciosPanes[0], stock: stockPanes[0], img: imgPanes[0]
        { id: 1, nombre: nombresPanes[1], precio: preciosPanes[1], stock: stockPanes[1], img: imgPanes[1]
        { id: 2, nombre: nombresPanes[2], precio: preciosPanes[2], stock: stockPanes[2], img: imgPanes[2]
        { id: 3, nombre: nombresPanes[3], precio: preciosPanes[3], stock: stockPanes[3], img: imgPanes[3]
        { id: 4, nombre: nombresPanes[4], precio: preciosPanes[4], stock: stockPanes[4], img: imgPanes[4]
    ],
    pasteles: [
        { id: 5, nombre: "Pastel de Fresa", precio: 175.00, stock: 20, img: "p1.jfif" },
        { id: 6, nombre: "Chocoflan", precio: 120.00, stock: 36, img: "p2.jfif" },
        { id: 7, nombre: "Moka", precio: 150.00, stock: 35, img: "p3.jpg" },
        { id: 8, nombre: "Frutos Rojos", precio: 190.00, stock: 23, img: "p4.jfif" },
        { id: 9, nombre: "Vainilla", precio: 100.00, stock: 20, img: "p5.jpg" }
    ],
    postres: [
        { id: 10, nombre: "Dona Chocolate", precio: 7.00, stock: 80, img: "ps1.avif" },
        { id: 11, nombre: "Pie de Limón", precio: 15.00, stock: 30, img: "ps2.jpg" },
        { id: 12, nombre: "Relámpago", precio: 12.00, stock: 50, img: "ps3.jfif" },
        { id: 13, nombre: "Cupcake", precio: 10.00, stock: 75, img: "ps4.jpg" },
        { id: 14, nombre: "Milhojas", precio: 8.00, stock: 45, img: "ps5.jpg" }
```

1. **Gestión de Datos y Memoria**
   - **Vectores y Objeto Inventario:** Actúan como la **Base de Datos** del programa. Los vectores iniciales alimentan al objeto inventario, que organiza la información en categorías (panes, pasteles, postres) para facilitar su acceso.

HTML5

CSS

```
function mostrarCategoria(cat) {
    const contenedor = document.getElementById('lista-productos');
    contenedor.innerHTML = '';
    inventario[cat].forEach(prod => {
        contenedor.innerHTML += `
            <div class="card">
                <img src="${prod.img}" alt="${prod.nombre}">
                <h4>${prod.nombre}</h4>
                <p>Precio: Q${prod.precio.toFixed(2)}</p>
                <p>Stock: <span id="stock-${prod.id}">${prod.stock}</span></p>
                <input type="number" id="cant-${prod.id}" min="1" max="${prod.stock}" value="1" style="width: 50px;">
                <button onclick="agregarAlCarrito('${cat}', ${prod.id})">Agregar</button>
            </div>
        `;
    });
```

2. Capa de Visualización (Frontend)
   - mostrarCategoria(cat): Su función es el Renderizado Dinámico. Toma los datos del inventario y "dibuja" las tarjetas de producto en el navegador. Es la ventana que el cliente ve y usa.

HTML 5

```
function agregarAlCarrito(cat, id) {
    const prod = inventario[cat].find(p => p.id === id);
    const cantInput = document.getElementById(`cant-${id}`);
    const cant = parseInt(cantInput.value);

    if (cant > prod.stock || cant <= 0) return alert("Cantidad no válida o insuficiente stock");

    const itemEnCarrito = carrito.find(item => item.id === id);
    if (itemEnCarrito) {
        if (itemEnCarrito.cantidadSeleccionada + cant > prod.stock) return alert("Supera el stock disponibl
        itemEnCarrito.cantidadSeleccionada += cant;
    } else {
        carrito.push({ ...prod, cantidadSeleccionada: cant, categoria: cat });
    }
    renderizarVenta();
}

function renderizarVenta() {
    const detalle = document.getElementById('detalle-venta');
    const totalSeccion = document.getElementById('total-seccion');
    if (carrito.length === 0) {
        detalle.innerHTML = "<p>El carrito está vacío.</p>";
        totalSeccion.innerHTML = "";
        return;
    }

    let subtotal = 0;
    let htmlBusqueda = "<ul>";
    carrito.forEach(item => {
        let filaTotal = item.precio * item.cantidadSeleccionada;
        subtotal += filaTotal;
        htmlBusqueda += `<li>${item.nombre} (x${item.cantidadSeleccionada}) - Q${filaTotal.toFixed(2)}</li>
    });
    htmlBusqueda += "</ul>";
    detalle.innerHTML = htmlBusqueda;

    let descuento = subtotal > 200 ? subtotal * 0.05 : 0;
    let totalFinal = subtotal - descuento;

    totalSeccion.innerHTML = `
        <div style="background: #f1f8e9; padding: 10px; border-radius: 5px;">
            <p>Subtotal: Q${subtotal.toFixed(2)}</p>
            <p style="color: green;">Descuento (5%): - Q${descuento.toFixed(2)}</p>
            <hr>
            <h3>Total a Pagar: Q${totalFinal.toFixed(2)}</h3>
        </div>
```

JS

3. Lógica de Negocio y Control
   - agregarAlCarrito(cat, id): Es el Controlador de Selección. Gestiona qué quiere el cliente, verifica que haya pan disponible en el estante (stock) y lo guarda temporalmente en la lista de compras.
   - renderizarVenta(): Funciona como el Monitor en Tiempo Real. Calcula subtotales y aplica automáticamente la lógica del descuento del 5% cuando la compra supera los Q200.

HTML5

```
function validarNIT() {
    const nitInput = document.getElementById('nitC');
    const errorMsg = document.getElementById('error-nit');
    const nit = nitInput.value.trim().toUpperCase();
    const regexNIT = /^[0-9]+-?[0-9kK]$/;

    if (nit === "CF" || regexNIT.test(nit)) {
        errorMsg.style.display = "none";
        nitInput.style.borderColor = "#ccc";
        return true;
    } else {
        errorMsg.style.display = "block";
        nitInput.style.borderColor = "red";
        return false;
    }
}

function procesarCompra() {
    const nombre = document.getElementById('nombreC').value;
    const nit = document.getElementById('nitC').value;
    const dir = document.getElementById('dirC').value;

    if (!nombre || !nit || carrito.length === 0) {
        return alert("Por favor complete los datos y agregue productos.");
    }

    if (!validarNIT()) return alert("NIT inválido.");
```

JS

4. Seguridad y Validación
   - validarNIT(): Es el Filtro de Integridad. Antes de permitir el cierre de la venta, asegura que la identificación del cliente (NIT o CF) cumpla con el formato legal de Guatemala, evitando errores en la facturación.

HTML 5

JS

```
// 1. Restar al vector para actualizar inventario
carrito.forEach(item => {
    const productoOriginal = inventario[item.categoria].find(p => p.id === item.id);
    productoOriginal.stock -= item.cantidadSeleccionada;
});

// 2. Calcular totales finales
let subtotal = carrito.reduce((acc, item) => acc + (item.precio * item.cantidadSeleccionada), 0)
let descuento = subtotal > 200 ? subtotal * 0.05 : 0;
let totalFinal = subtotal - descuento;

// 3. USAR document.write en una nueva ventana para mostrar la información solicitada
const ventanaFactura = window.open('', '_blank');
ventanaFactura.document.write(`
    <html>
    <head><title>Factura - Todo En Panes</title></head>
    <body style="font-family: Arial; padding: 20px; border: 1px solid #000;">
        <h1>Panadería - Todo En Panes</h1>
        <hr>
        <h3>Datos del Cliente</h3>
        <p><strong>Nombre:</strong> ${nombre}</p>
        <p><strong>NIT:</strong> ${nit}</p>
        <p><strong>Dirección:</strong> ${dir}</p>
        <hr>
        <h3>Detalle de Compra</h3>
        <ul>
            ${carrito.map(i => `<li>${i.nombre} x ${i.cantidadSeleccionada} - Q${(i.precio * i.c
        </ul>
        <p>Subtotal: Q${subtotal.toFixed(2)}</p>
        <p>Descuento Realizado: Q${descuento.toFixed(2)}</p>
        <h2>Total Pagado: Q${totalFinal.toFixed(2)}</h2>
        <button onclick="window.print()">Imprimir Recibo</button>
    </body>
    </html>
`);
ventanaFactura.document.close();

// Resetear interfaz principal
carrito = [];
renderizarVenta();
mostrarCategoria('panes');
alert("Venta procesada e inventario actualizado.");
```

5. Finalización y Salida (Output)
   - procesarCompra(): Es el Cierre de Ciclo. Realiza tres acciones críticas finales:
     a. Sincronización: Resta físicamente los productos del inventario.
     b. Generación: Utiliza document.write para crear el reporte legal en una nueva pestaña.
     c. Reseteo: Limpia el sistema para recibir al siguiente cliente

GRACIAS POR SU ATENCION