Documentación del Código de la Aplicación de Compras

Introducción:

Este documento proporciona una guía detallada sobre la estructura y funcionalidades del código de la aplicación de compras en línea. La aplicación está diseñada para ofrecer a los usuarios una experiencia interactiva y amigable mientras exploran y adquieren productos de diferentes categorías. La explicación se centra en el código JavaScript (script.js) que impulsa la lógica de la interfaz de usuario, la gestión del carrito de compras y la interacción con el usuario.

A lo largo de esta documentación, se explorarán los siguientes aspectos clave del código:

Inicio de la Aplicación:

El script se inicia con un evento que garantiza la ejecución del código después de que el contenido HTML se haya cargado por completo.

Obtención de Datos:

Se describen los pasos para recuperar la lista completa de productos desde el almacenamiento de sesión, esencial para la presentación dinámica de productos.

Interacción con el DOM:

Se explican las selecciones de elementos del DOM para permitir la manipulación y visualización de datos en la interfaz de usuario.

Identificación de la Categoría de Productos:

Se muestra cómo el código identifica la categoría de productos basándose en la ruta actual, personalizando así la visualización de productos.

Renderizado Dinámico de Productos:

Se analiza la función encargada de crear y presentar dinámicamente los elementos HTML de productos en la página principal.

Manejo de Eventos y Modal:

Se detallan las funciones responsables de gestionar eventos de clic en botones, abrir y cerrar modales, y proporcionar información adicional sobre los productos.

Lógica del Carrito de Compras:

Se exploran funciones cruciales para el manejo del carrito de compras, incluido el agregado de productos, visualización del carrito y acciones sobre productos en el carrito.

Búsqueda y Filtrado:

Se explica la funcionalidad de búsqueda que permite a los usuarios filtrar productos según sus preferencias.

Navegación entre Categorías:

Se presentan eventos de clic en botones de navegación, permitiendo a los usuarios explorar distintas categorías y regresar a la página de inicio.

Este documento está diseñado para ofrecer una comprensión clara y detallada del código subyacente a la aplicación de compras en línea, facilitando su mantenimiento y posibles mejoras en el futuro.

Código: categorias.js

Este archivo JavaScript, llamado categorias.js, se centra en la funcionalidad y visualización de la categoría de productos en una tienda en línea. A continuación, se proporciona una explicación más detallada de algunas de las funciones clave presentes en el código:

Estructura del Código:

javascript

```
document.addEventListener("DOMContentLoaded", function () {
  // ... (variables y constantes)

  // Determinar el identificador según la ruta
  let identificador;
  // ... (lógica para determinar el identificador según la ruta)

  // Filtrar la lista de productos según el identificador
  let clothesList = clothesListComplete.filter(function (producto) {
    return producto.identificador === identificador;
  });

  // Renderizar la lista de productos en la interfaz
  renderClothes(clothesList);

  // ... (más funciones y eventos)
});
```

Funciones Principales:

1. renderClothes(clothes)

Esta función es esencial para generar dinámicamente la representación visual de los productos en la interfaz. Utiliza el contenido de la lista de productos para crear tarjetas de productos y las agrega al contenedor correspondiente en el HTML.

javascript

```
function renderClothes(clothes) {
  // Limpia el contenedor de productos
  clothesContainer.innerHTML = "";


  clothes.forEach((response) => {
    // Crea dinámicamente la estructura de la tarjeta de producto
    const cardClothesElement = document.createElement("div");
    cardClothesElement.innerHTML = `
      <!-- Estructura de la tarjeta de producto -->
    `;

    // Agrega eventos a los botones en la tarjeta de producto

    clothesContainer.appendChild(cardClothesElement);
  });
}
```

2. openModalWithDetails(product)

Esta función se encarga de mostrar detalles específicos de un producto en un modal. Además, permite agregar el producto al carrito cuando se hace clic en el botón correspondiente.

javascript

```
function openModalWithDetails(product) {
  // Obtiene el modal de detalles y establece su contenido
  const modal = document.getElementById("ModalDetails");
  modal.innerHTML = `
    <!-- Estructura del modal de detalles -->
  `;

  // Agrega eventos a los botones en el modal

  // Muestra el modal
  modal.style.display = "block";
}
```

3. closeModal()

Cierra el modal cuando se hace clic en el botón de cierre.

javascript

```
function closeModal() {
  const modal = document.getElementById("ModalDetails");
  modal.style.display = "none";
}
```

4. exportToJson(data)

Esta función exporta el producto actual a formato JSON y lo agrega al carrito de compras.

javascript

```
function exportToJson(data) {
  // Convierte el objeto a formato JSON y lo agrega al carrito
  const jsonData = JSON.stringify(data);
  addToCart(data);
}
```

Funciones del Carrito:

El código también incluye funciones relacionadas con la gestión del carrito de compras, como addToCart, removeProduct, decreaseQuantity, showModal, buyProducts, y otras. Estas funciones controlan cómo se manipulan y muestran los productos en el carrito de compras.

Además, hay funciones relacionadas con la navegación entre páginas y eventos de botones (btnHome, btnMan, btnWoman, btnGirl, btnBoy), que facilitan la transición del usuario entre las diferentes secciones de la tienda.

Código: clothes.js

El archivo clothes.js contiene la lista de productos para la tienda en línea. A continuación, se proporciona una explicación detallada de las funciones y la estructura del código:

1. Estructura del Código:

javascript

```
const clothesList = [
  // ... (array de objetos representando productos)
];


export { clothesList };
```

2. Productos:

Cada objeto en clothesList describe un producto específico, incluyendo su nombre, descripción, precio, material, imagen, cantidad en stock e identificador de categoría (MAN, WOMAN, KIDS, GIRL).

3. Ejemplos de Productos:

Producto 1: Camiseta Deportiva

javascript

```
{
  "id": 1,
  "nombreProducto": "Camiseta Deportiva",
  // ... (otras propiedades)
}
```

Producto 2: Pantalones

javascript

```
{
  "id": 2,
  "nombreProducto": "Pantalones",
  // ... (otras propiedades)
}
```

4. Exportación del Módulo:

El módulo exporta la lista de productos (clothesList) para su uso en otros archivos del proyecto.

javascript

```
export { clothesList };
```

Este archivo sirve como fuente de datos centralizada para los productos, facilitando su acceso desde otras partes del proyecto.

Este módulo es esencial para mantener la información de los productos de manera organizada y accesible desde otros archivos que necesiten acceder a la lista de productos.

Código: facturar.js

Este archivo JavaScript, denominado facturar.js, se encarga de generar una factura para los productos en el carrito de compras. A continuación, se explica en detalle cada parte del código:

1. Evento "DOMContentLoaded":

javascript

```
document.addEventListener("DOMContentLoaded", function () {
  // ... (resto del código)
});
```

Este evento se dispara cuando el documento HTML ha sido completamente cargado y parseado. Es el punto de inicio de la ejecución del código.

2. Obtención de Productos del Carrito:

javascript

```
const product = JSON.parse(sessionStorage.getItem('productsInCart'));
```

Se obtienen los productos del carrito almacenados en la sesión del navegador.

3. Generación de la Factura:

javascript

```
const invoiceBody = document.getElementById('invoiceBody');
```

```
let totalAmount = 0;

product.forEach(producto => {
  // ... (resto del código)
});
```
Se recorren los productos del carrito para calcular el subtotal de cada uno y actualizar el monto total. Además, se crea dinámicamente una fila en la tabla de la factura para cada producto.

4. Actualización del Total y Exportación:

javascript

```
document.getElementById('totalAmount').textContent = totalAmount.toFixed(2);

document.getElementById('exportButton').addEventListener('click', function() {
  // ... (resto del código)
});
```
Se actualiza el elemento HTML que muestra el total de la factura. Además, se agrega un evento al botón de exportación que abre una nueva ventana y muestra la factura para su impresión.

5. Configuración de la Nueva Ventana:

javascript

```
var invoiceHtml = document.getElementById('invoiceDiv').innerHTML;

var newWindow = window.open('', '');
newWindow.document.head.innerHTML = `
  <script src="js/facturar.js"></script>
  <link rel="stylesheet" href="css/general.css">
  <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@422&display=swap"
rel="stylesheet">
`;

newWindow.document.body.innerHTML = invoiceHtml;
setTimeout(() => {
  newWindow.print();
}, 1100);
```
Se crea una nueva ventana, se copian los estilos y scripts necesarios, y se imprime automáticamente el contenido de la factura.

6. Botón de Regreso:

javascript

```
document.getElementById("returnButton").addEventListener("click", function() {
  window.history.back();
});
```
Se agrega un evento al botón de regreso que lleva a la página anterior en la historia del navegador.

Este archivo se encarga de presentar de manera efectiva la información de la factura y proporcionar opciones de impresión y regreso.

Código: script.js

Este archivo JavaScript, llamado script.js, se encarga de manejar la lógica principal de la aplicación de carrito de compras. A continuación, se detallan los elementos clave del código:

1. Importación de Datos:

javascript

```
import { clothesList } from "../js/clothes.js";
```
Se importa la lista de productos desde el archivo clothes.js.

2. Evento "DOMContentLoaded":

javascript

```
document.addEventListener("DOMContentLoaded", function () {
  // ... (resto del código)
});
```
Este evento se dispara cuando el documento HTML ha sido completamente cargado y parseado, iniciando la ejecución del código.

3. Renderizado de Productos:

javascript

```
renderClothes(clothesList);
```

La función renderClothes se encarga de generar dinámicamente la presentación de los productos en la interfaz gráfica, incluyendo botones para agregar y ver detalles.

4. Manejo de Búsqueda:

javascript

```
searchInput.addEventListener("input", () => {
  // ... (resto del código)
});
```

Se añade un evento de escucha para el campo de búsqueda, filtrando los productos en tiempo real según el término ingresado.

5. Agregar a Carrito y Ver Detalles:

javascript

```
const exportBtn = cardClothesElement.querySelector(".export-btn");
exportBtn.addEventListener("click", () => {
  exportToJson(response);
  cartNumberProducts(shopCart.length);
  sessionStorage.setItem("productsInCart", JSON.stringify(shopCart));
});

const moreBtn = cardClothesElement.querySelector(".more-btn");
moreBtn.addEventListener("click", () => {
  openModalWithDetails(response);
});
```

Los botones de agregar al carrito y ver más están asociados a las funciones exportToJson y openModalWithDetails, respectivamente.

6. Lógica del Carrito:

javascript

```
let shopCart = [];
// ... (resto del código)

function addToCart(product) {
  // ... (resto del código)
}

function showModal(products) {
  // ... (resto del código)
}

function removeProduct(productId) {
  // ... (resto del código)
}
```

```
function decreaseQuantity(productId) {
  // ... (resto del código)
}

function buyProducts(products) {
  // ... (resto del código)
}
```
Estas funciones gestionan la manipulación del carrito, como agregar, mostrar, eliminar y disminuir la cantidad de productos.

7. Manejo de Eventos y Navegación:

javascript

```
// ... (resto del código)

let btnMan = document.getElementById("btnMan");
let btnWoman = document.getElementById("btnWoman");
let btnGirl = document.getElementById("btnGirl");
let btnBoy = document.getElementById("btnBoy");

btnMan.addEventListener("click", function () {
  window.location.href = "hombres.html";
});

btnWoman.addEventListener("click", function () {
  window.location.href = "mujeres.html";
});

btnGirl.addEventListener("click", function () {
  window.location.href = "ninas.html";
});

btnBoy.addEventListener("click", function () {
  window.location.href = "ninos.html";
});
```
Se añade funcionalidad a los botones de navegación, redirigiendo a diferentes secciones de la tienda al hacer clic.

Este archivo centraliza la lógica del carrito, desde el renderizado de productos hasta el manejo de eventos y la gestión del carrito de compras.

Resumen:

Estos cuatro archivos JavaScript están diseñados para funcionar en conjunto y formar una aplicación de comercio electrónico. Aquí hay un resumen de cómo interactúan:

- **clothes.js:** Contiene la información detallada de una lista de productos, organizada por categorías (hombres, mujeres, niños, niñas). Exporta esta lista para ser utilizada en otros archivos.

- **script.js:** Este archivo principal maneja la lógica de la aplicación. Utiliza la lista de productos de clothes.js para renderizar dinámicamente tarjetas de productos en la interfaz gráfica. Además, gestiona eventos como la búsqueda de productos, la apertura de detalles de productos y la interacción con el carrito de compras. También controla la navegación entre secciones de la tienda.

- **modal.js:** Proporciona funciones para abrir y cerrar un modal que muestra detalles específicos de un producto. Se utiliza en script.js para mostrar más información sobre un producto seleccionado.

- **facturar.js:** Se ejecuta al cargar la página de facturación. Utiliza la información almacenada en la sesión sobre los productos seleccionados para generar una factura dinámica, calculando el subtotal de cada producto y el monto total. Permite exportar la factura como un documento imprimible.

En conjunto, estos archivos forman una aplicación cohesiva donde script.js maneja la interacción del usuario, utiliza la información de clothes.js para mostrar productos y detalles, y facturar.js utiliza los datos del carrito para generar una factura. modal.js complementa proporcionando una visualización detallada de los productos. La modularidad de estos archivos facilita la mantenibilidad y expansión del sistema.

FIGMA

https://www.figma.com/file/8hsVyEkeMbbxas2deUxH06/Shop-project?type=design&node-id=0%3A1&mode=design&t=Pu7YRgMQ4KeBCTmp-1