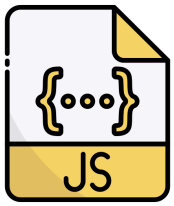
Clase 10

Sitio: Talento Tech
Curso: Desarrollo Web 2_11
Libro: Clase 10

Imprimido por: Ailen Callizaya
Día: martes, 19 de noviembre de 2024, 15:11

Tabla de contenidos

1. Clase N°10



¿Qué vamos a trabajar en esta clase?

- Repaso clases 6, 7, 8 y 9

Resultados de aprendizaje:

- Conocer cómo implementar un carrito de compras en nuestro E-Commerce
- Repasar todos los contenidos que se aprendieron a lo largo de la cursada

2. Creando nuestro E-Commerce

El objetivo de esta clase es terminar de armar nuestro E-commerce con todos los conceptos que fuimos aprendiendo durante la cursada.

Abajo les dejo el paso a paso para crear un carrito de compras con una breve explicación de cada uno de ellos.

1. Declaración de una variable global “cartItems”:

```
let cartItems = [];
```

Declaramos una variable llamada “cartItems”, que será utilizada para almacenar los elementos agregados al carrito de compras. Se inicializa como un array vacío.

2. Obtención de referencias a elementos del DOM:

```
const cartButton = document.getElementById('cartButton');
const cartModal = document.getElementById('cartModal');
const cartItemsContainer = document.getElementById('cartItems');
```

En estas tres líneas de código estamos obteniendo referencias a elementos del DOM que necesitaremos para interactuar con el carrito de compras y el modal asociado.

3. Event listener para abrir el modal del carrito:

```
// Función para abrir el modal al hacer clic en el balón del carrito
cartButton.addEventListener('click', () => {
    updateCartDisplay(); // Actualizar la visualización del carrito antes de abrir el modal
    cartModal.style.display = 'block';
});
```

Este bloque de código agrega un event listener al botón del carrito. Cuando se hace clic en este botón, se ejecuta una función que actualiza la visualización del carrito y muestra el modal.

4. Función para cerrar el modal al hacer clic en la "x" de cierre:

```
// Función para cerrar el modal al hacer clic en la "x" de cierre
document.querySelector('.close').addEventListener('click', () => {
    cartModal.style.display = 'none';
});
```

Esta parte del código agrega un event listener al elemento con la clase .close, que representa el icono de cierre en el modal del carrito. Cuando se hace clic en este icono, se ejecuta la función de callback que establece el estilo display del modal en "none", lo que hace que el modal se cierre y desaparezca de la vista.

5. Función para cerrar el modal al hacer clic fuera del contenido del modal:

```
// Función para cerrar el modal al hacer clic fuera del contenido del modal
window.addEventListener('click', event => {
    if(event.target == cartModal) {
        cartModal.style.display = 'none';
    }
})
```

Acá agregamos un event listener a la ventana. Cuando se hace clic en cualquier parte de la ventana, esta función comprueba si el objetivo del clic es el modal del carrito (cartModal). Si es así, el modal se cierra de la misma manera que en el punto anterior.

6. Función para agregar productos al carrito:

```
function addToCart(button, title, price) {
    // Verificar si el producto ya está en el carrito
    const existingItem = cartItems.find(item => item.title === title);
if (existingItem) {
    existingItem.quantity++; // Incrementar la cantidad si el producto ya está en el carrito
```

```
} else {
    cartItems.push({ title, price, quantity: 1 }); // Agregar el producto al carrito con cantidad 1
}
    updateCartDisplay();
    updateCartCount(); // Llamar a la función para actualizar el número de productos en el carrito en la barra de navegación
    // Cambiar el texto y el color del botón y agregar la clase "added"
    button.textContent = 'Agregado';
}
```

Esta función se llama cuando se hace clic en el botón "Agregar al carrito" de un producto. Agregando el producto al array cartItems, incrementando su cantidad si ya está en el carrito, o agregándolo con una cantidad de 1 si es nuevo.

7. Función para actualizar el botón de agregar al carrito:

```
function updateCartButton() {
    const addButton = document.getElementById('add-to-cart-button');
    addButton.textContent = 'Agregado';
    addButton.classList.remove('btn-primary');
    addButton.classList.add('btn-success');
}
```

Esta función se encarga de cambiar el texto y el color del botón de "Agregar al carrito" después de que un usuario haya agregado un producto al carrito. Primero, encuentra el botón por su ID, luego cambia el texto a "Agregado" y actualiza las clases CSS del botón para que tenga el color verde que indica que el producto se ha agregado con éxito.

8. Función para actualizar el contador de productos en el carrito:

```
function updateCartCount() {
    const cartCountElement = document.getElementById('cartCount');
    const totalCount = cartItems.reduce((total, item) => total + item.quantity, 0);
    cartCountElement.textContent = totalCount;
}
```

En esta función, se calcula y actualiza el número total de productos en el carrito. Utiliza el método reduce() para sumar las cantidades de todos los productos en el array cartItems. Luego, actualiza el contenido del elemento HTML que muestra el contador con el número total de productos.

9. Función para remover un producto del carrito:

```
function removeFromCart(title) {
    cartItems = cartItems.filter(item => item.title !== title);
    updateCartDisplay();
    updateCartCount(); // Llamar a la función para actualizar el número de productos en el carrito en la barra de
```

```
navegación
}
```

Definimos una función para eliminar un producto específico del carrito de compras. Utiliza el método **filter()** para crear un nuevo array que excluya el producto con el título especificado. Luego, actualiza la visualización del carrito y el contador de productos llamando a las funciones correspondientes.

10. Función para actualizar la visualización del carrito:

```
// Función para actualizar la visualización del carrito
function updateCarteDisplay() {
    const cartItemsContainer = document.getElementById('cartItems');
    cartItemsContainer.innerHTML = ''; // Limpiar el contenido anterior
    let total = 0;
    carteItems.forEach(item => {
        const itemTotal = item.price * item.quantity;
        total += itemTotal;
        const cartItemElement = document.createElement('div');
        cartItemElement.classList.add('cart-item');
        cartItemElement.innerHTML = `
    <p>${item.title} --> $${item.price} = $${itemTotal}</p>
    <div class="quanty-controls">
        <button class="quantity-btn" onclick="decrementQuantity('${item.title}')"></button>
        <span class="quantity">${item.quantity}</span>
        <button class="quantity-btn" onclick="incrementQuantity('${item.title}')"></button>
    </div>
    <button class="remove" onclick="removeFromCart('${item.title}')">Remove</button>
```

```
        `;
            cartItemsContainer.appendChild(cartItemElement);
        });
        const cartTotalElement = document.getElementById('cartTotal');
        cartTotalElement.textContent = `$${total.toFixed(2)}`;
    }
```

Esta función actualiza la visualización del carrito en el modal. Itera sobre los elementos en cartItems, crea un elemento HTML para cada uno y lo agrega al contenedor del carrito. También muestra el total de la compra.

11. Inicialización de la variable currentQuantity:

```
//Variables globales
let currentQuantity = 1; // Inicialmente, la cantidad es 1
```

Esta línea inicializa una variable llamada currentQuantity con un valor de 1. Esta variable se utiliza para llevar el seguimiento de la cantidad de un producto en particular que un usuario desea agregar al carrito.

12. Obtención de referencias a elementos del DOM para la gestión de la cantidad de productos:

```
// Obtener referencias a los elementos del DOM
const decrementButton = document.getElementById('decrementQuantity');
```

```
const incrementButton = document.getElementById('icrementQuantity');
const quantityElement = document.getElementById('quantity');
// const buyButton = document.getElementById('buyButton');
```

Estas líneas de código obtienen referencias a los elementos del DOM que se utilizan para la gestión de la cantidad de productos en el carrito. **decrementButton** e **incrementButton** representan los botones de decremento e incremento respectivamente, mientras que **quantityElement** representa el elemento que muestra la cantidad actual.

13. Funciones para incrementar y decrementar la cantidad de productos en el carrito:

```
function incrementQuantity(title) {
    const item = cartItems.find(item => item.title === title);
   if (item) {
    item.quantity++;
    updateCartDisplay();
    updateCartCount();
   }
}
function decrementQuantity(title) {
    const item = cartItems.find(item => item.title === title);
    if (item && item.quantity > 1) {
        item.quantity--;
        updateCartDisplay();
        updateCartCount();
    }
}
```

Estas funciones se utilizan para aumentar o disminuir la cantidad de un producto en el carrito. Actualizan la visualización del carrito y el contador de productos en la barra de navegación.

14. Event listener para el botón de incremento de cantidad:

```
incrementButton.addEventListener('click', () => {
    currentQuantity++;
    updateQuantityDisplay();
});
```

Aquí se agrega un event listener al botón de incremento de cantidad. Cuando el usuario hace clic en este botón, se incrementa el valor de currentQuantity en 1 y luego se actualiza la visualización de la cantidad llamando a la función updateQuantityDisplay().

15. Función para actualizar la visualización de la cantidad:

```
// Función para actualizar la visualización de la cantidad
function updateQuantityDisplay() {
    quantityElement.textContent = currentQuantity;
}
```

Esta función se encarga de actualizar la visualización de la cantidad en la interfaz del carrito. Actualiza el contenido del elemento HTML que muestra la cantidad actual con el valor de **currentQuantity**.

3. ? Pregunta de reflexión

En el contexto de las Promesas:

¿Puedes pensar en una situación específica en un E-commerce donde sería beneficioso utilizar promesas para cargar datos de productos de manera asíncrona?

¿Cómo crees que la implementación de un carrito de compras con JavaScript y eventos mejora la experiencia del usuario en un sitio web de comercio electrónico?

4. Desafío #10

¡Entrega final!

¡Felicitaciones por haber llegado hasta acá!

Esta vez te toca entregar tu propio **e-commerce** personalizado donde podrán darle funcionalidad e interacción a su página web.

Sean libres de agregar lo que consideren necesario y de cambiar el código, sus id, variables, etc. para darle su propio estilo.