DRAG AND DROP API SHOPPING CART

Set up data and DOM elements. Start with `shoppingcart.html` and `cart.css`, we will create `cart.js` in the `js` folder. Use images from the `img` folder.

- `const productsData = [ ... ]`: This line declares a constant variable named `productsData` and assigns it an array containing product objects. Each object has properties for `id`, `name`, `price`, and `image` (presumably a path to the product image).
- `document.addEventListener('DOMContentLoaded', () => { ... });`: This line sets up an event listener for the `DOMContentLoaded` event. When the DOM (Document Object Model) is fully loaded and parsed, the provided function is executed.
    - Inside the function, several constant variables are declared using `const`:
        - `productsContainer`: This stores a reference to the HTML element with the class `products` (likely the container for displaying product listings).
        - `shoppingCart`: This stores a reference to the HTML element with the class `shopping-cart` (likely the element representing the shopping cart).
        - `shoppingCartList`: This stores a reference to the HTML element with the class `shopping-cart-list` (likely where the list of items in the cart is displayed).
        - `productQuantity`: This stores a reference to the HTML element with the class `product-quantity` (likely where the total quantity of items in the cart is displayed).
        - `totalPrice`: This stores a reference to the HTML element with the class `total-price` (likely where the total price of all items in the cart is displayed).
        - `emptyCartBtn`: This stores a reference to the HTML element with the class `empty-cart-btn` (likely a button for clearing the shopping cart).

```
const productsData = [
  { id: 1, name: "JavaScript and JQuery: Interactive Front-End
Web Development", price: 10, image: "img/product1.jpg" },
  { id: 2, name: "Eloquent JavaScript", price: 15, image: "img/
product2.jpg" },
  { id: 3, name: "You Don't Know JS", price: 20, image: "img/
product3.jpg" },
  { id: 4, name: "Effective JavaScript", price: 10, image: "img/
product4.jpg" },
```

```
  { id: 5, name: "JavaScript: The Good Parts", price: 15, image:
"img/product5.jpg" },
  { id: 6, name: "Deep JavaScript: Theory and Techniques",
price: 20, image: "img/product6.jpg" },

  ];

document.addEventListener('DOMContentLoaded', () => {
// DOM elements
const productsContainer = document.querySelector('.products');
const shoppingCart = document.querySelector('.shopping-cart');
const shoppingCartList = document.querySelector('.shopping-cart-
list');
const productQuantity = document.querySelector('.product-
quantity');
const totalPrice = document.querySelector('.total-price');
const emptyCartBtn = document.querySelector(`.empty-cart-btn');
```

2. Cart items and initial rendering:

- `let cartItems = [];`: This line declares a variable named `cartItems` as an empty array. This will be used to store product objects added to the shopping cart.
- `renderProducts()`: This line calls the `renderProducts` function (explained later) to populate the product listings initially.
- `renderCartItems()`: This line calls the `renderCartItems` function (explained later) to display the initial state of the shopping cart (likely empty).

```
let cartItems = [];
function renderProducts() {
  productsContainer.innerHTML = '';
  productsData.forEach(product => {
    const productElement = document.createElement('div');
    productElement.classList.add('product');
    productElement.draggable = true;
    productElement.dataset.productId = product.id;
    productElement.innerHTML = `
      <img src="${product.image}" alt="${product.name}"><br>
      <h3>${product.name}</h3>
      <p>$${product.price}</p>
    `;
   productsContainer.appendChild(productElement);

    // Drag event listeners
    productElement.addEventListener('dragstart', dragStart);
  });
}
```

3. Rendering product listings:

- `function renderProducts() { ... }`: This function defines the logic for rendering product listings:
  - `productsContainer.innerHTML = '';`: This line clears any existing content within the `productsContainer` element.
  - It iterates through the `productsData` array using `forEach`. For each product:
    - It creates a new `div` element and sets its class to `product`.
    - It sets the `draggable` attribute to `true` to enable dragging products.
    - It sets the `data-productId` attribute to the product's `id` for easy identification during drag and drop.
    - It creates a string containing the product's image, name, and price using template literals (```) for better readability.
    - It sets the `innerHTML` property of the created `div` element with the product information string.
    - Finally, it appends the created product element to the `productsContainer`.
  - Inside the loop, it also adds an event listener for the `dragstart` event on each product element. This calls the `dragStart` function (explained later) to handle drag initiation.

```
function renderCartItems() {
  shoppingCartList.innerHTML = '';
  cartItems.forEach(item => {
    const cartItemElement = document.createElement('li');
    cartItemElement.innerHTML = `
      ${item.name} - $${item.price} x ${item.quantity}
      <button class="remove-item" data-product-id="$
{item.id}">Remove</button>
    `;
    shoppingCartList.appendChild(cartItemElement);

    cartItemElement.querySelector('.remove-
item').addEventListener('click', removeItem);
  });
```

4. Rendering cart items:

- `function renderCartItems() { ... }`: This function defines the logic for rendering the list of items in the shopping cart:
  - `shoppingCartList.innerHTML = '';`: This line clears any existing content within the `shoppingCartList` element.

- It iterates through the `cartItems` array using `forEach`. For each item:
  - It creates a new `li` (list item) element.
  - It creates a string containing the item's name, price, quantity, and a "Remove" button with a `data-product-id` attribute set to the item's `id`.
  - It sets the `innerHTML` property of the created `li` element with the cart item information string.
  - It appends the created list item element to the `shoppingCartList`.
- After iterating through cart items, it retrieves the "Remove" button element within each list item using `querySelector`.
- It adds an event listener for the `click` event on each "Remove" button. This calls the `removeItem` function (explained later) to handle removing items from the cart.
- It updates the displayed product quantity and total price.

```
function renderCartItems() {
  shoppingCartList.innerHTML = '';
  cartItems.forEach(item => {
    const cartItemElement = document.createElement('li');
    cartItemElement.innerHTML = `
      ${item.name} - $${item.price} x ${item.quantity}
      <button class="remove-item" data-product-id="$
{item.id}">Remove</button>
    `;
    shoppingCartList.appendChild(cartItemElement);

    cartItemElement.querySelector('.remove-
item').addEventListener('click', removeItem);
  });

  // Update cart summary
  productQuantity.textContent = cartItems.reduce((sum, item) =>
sum + item.quantity, 0);
  const total = cartItems.reduce((sum, item) => sum + item.price
* item.quantity, 0);
  totalPrice.textContent = `$${total.toFixed(2)}`;
}
```

Drag and Drop handlers:

- `function dragStart(event) { ... }`: This function handles the start of a drag operation:

    - It sets the data being transferred during the drag operation to the `data-productId` of the dragged element.
- `function dragOver(event) { ... }`: This function handles the `dragover` event, preventing default behavior (like dropping on an invalid target).
- `function drop(event) { ... }`: This function handles the `drop` event, which occurs when an element is dropped onto a valid target:
    - It prevents the default behavior.
    - It retrieves the `data-productId` from the data transfer.
    - It finds the product object with the corresponding ID from the `productsData` array.
    - It calls the `addToCart` function (explained later) to add the product to the cart.

```
function dragStart(event) {
  event.dataTransfer.setData('text/plain',
event.target.dataset.productId);
}

function dragOver(event) {
  event.preventDefault();
}

function drop(event) {
  event.preventDefault();
  const productId = event.dataTransfer.getData('text/plain');
  const product = productsData.find(p => p.id ===
parseInt(productId));
  addToCart(product);
}
```

6. Cart functions:

- `function addToCart(product) { ... }`: This function handles adding a product to the cart:
    - It checks if the product already exists in the cart using `findIndex`.
    - If the product exists, it increments its quantity.
    - If the product doesn't exist, it creates a new cart item object with the product details and a quantity of 1, and adds it to the `cartItems` array.
    - It calls the `renderCartItems` function to update the cart display.
- `function removeItem(event) { ... }`: This function handles removing a product from the cart:
    - It retrieves the product ID from the clicked "Remove" button.

    - It filters the `cartItems` array to remove the item with the specified ID.
    - It calls the `renderCartItems` function to update the cart display.
- `function emptyCart() { ... }`: This function empties the cart:
    - It sets the `cartItems` array to an empty array.
    - It calls the `renderCartItems` function to update the cart display.

```
function addToCart(product) {
  const existingItemIndex = cartItems.findIndex(item => item.id
=== product.id);


  if (existingItemIndex !== -1) {
    cartItems[existingItemIndex].quantity++;
  } else {
    cartItems.push({ ...product, quantity: 1 });
  }

  renderCartItems();
}

function removeItem(event) {
  const productId = parseInt(event.target.dataset.productId);
  cartItems = cartItems.filter(item => item.id !== productId);
  renderCartItems();
}

function emptyCart() {
  cartItems = [];
  renderCartItems();
}
```

7. Event listeners and initialization:

- `shoppingCart.addEventListener('dragover', dragOver);`: This adds an event listener to the `shoppingCart` element for the `dragover` event, calling the `dragOver` function.
- `shoppingCart.addEventListener('drop', drop);`: This adds an event listener to the `shoppingCart` element for the `drop` event, calling the `drop` function.
- `emptyCartBtn.addEventListener('click', emptyCart);`: This adds an event listener to the `emptyCartBtn` element for the `click` event, calling the `emptyCart` function.

- `renderProducts();`: This calls the `renderProducts` function to initially render the product listings.
- `renderCartItems();`: This calls the `renderCartItems` function to initially render the shopping cart (empty).

```
shoppingCart.addEventListener('dragover', dragOver);
shoppingCart.addEventListener('drop', drop);
emptyCartBtn.addEventListener('click', emptyCart);

// Initial render
renderProducts();
renderCartItems();
});
```

Test by opening `shoppingcart.html` and dragging items over to the cart, ***but not by image.***