Project: Shop Local Storage!

Learning how to set up an online shop is useful for everyone from large e-commerce companies to an indie small business owner with a hobby website. Shopping carts are an integral part of any online store. In this project you will set one up with JavaScript -- and then, sell, baby, sell!

As we previously reviewed, `localStorage` is an ideal place to store data in the browser without affecting the server-side database. One use-case for`localStorage` is saving a user's shopping cart so it persists from page to page. Now that you know how to set and get items in `localStorage`, put that knowledge to use in this project.

In this project, you will practice:

- Setting up a quick shopping cart
- Storing items in `localStorage`
- Reading items in `localStorage`
- Removing an item from `localStorage`

Project overview

We've set up a project folder called **local-storage-project.zip**. Use these files to complete the project.

We have filled out the HTML file with a few elements and the CSS with some basic styles to represent a page where users can add items to a shopping cart. Open **storage-project.html** in a browser to see what this page looks like.

In Phases 1-3, you will write JavaScript functions to let the user do the following:

- Add an item to the shopping cart (*i.e. store the item in* `localStorage`)
- Display the items in the shopping cart (*i.e. read the item in* `localStorage`)
- Remove an item from the shopping cart (*i.e. remove the item from* `localStorage`)

Phase 1: Write a function to store an item in the cart

Our HTML file contains a simple form with a couple of inputs and a submit button, as well as an area to display cart items.

In a code editor, open the **storage-project.js** file, where you'll write all of your JavaScript. You should see an event listener for `DOMContentLoaded` because, again, it's a good idea to wrap functions that manipulate the DOM in this listener (or else, have a bad time with async issues!).

The first order of business is to write a function that stores the form values in `localStorage` whenever the user clicks the `add-to-cart` button. This function should:

- Listen for a click event
- Grab the form values
- Store the item in `localStorage`

Hint: Recall that we use Storage.setItem() to set a key-value pair in `localStorage`.

Phase 2: Write a function to display the cart items

The second order of business is to write a function to display the items that have been saved in `localStorage` to the Shopping Cart part of the page. In your HTML file, this is represented by the `div` with an ID of `shopping-cart`. This function should:

- Retrieve the item(s) stored in `localStorage`
- Insert the items into the DOM in the `shopping-cart` DIV
- Display both the item's name and its quantity

Hint: You might want to use a loop!

Hint: Recall that we use Storage.getItem(key) to get a value via its key.

Hint: We can also use Storage.key(index) to get a key name via its index.

Phase 3: Write a function to remove items from the cart

The last thing we need to have a fully functional cart is for the user to be able to remove items from the cart. Write a function that lets the user remove items from their cart. In order to do this, you might want to add to or amend the function you wrote in Phase 2.

Insert **Remove** buttons next to each cart item you inserted on the page. Then, write a function that does the following:

- Listen for a click event on the "Remove" buttons
- Remove the corresponding item from `localStorage`
- Remove the item from the `shopping-cart` DIV

Hint: We can use Storage.removeItem() to remove an item from `localStorage`.

Hint: We can use document.querySelectorAll() to get a Node List of the buttons. Then, loop through the list.

Hint: Recall what you know about event.target and Element.parentNode. It would be helpful to store the item's key in an ID attribute on the Remove button or the button's containing DIV, so that you know which corresponding key-value pair to remove from `localStorage`.

Hint: We can use location.reload to refresh the page when an item is removed. If everything is working correctly, the other cart items should still display after a page refresh.

Check that your shopping cart functions correctly

After you've written your JS, test that the page functions as it should. A user should be able to add an item with a given quantity to the cart. You should be able to see this item in `localStorage` in your browser's Developer Tools, and it should show up in the Shopping Cart section of the page.

When a user removes an item by clicking the "Remove" button, that item should be removed from `localStorage` as well as the page.

When you refresh the page, or close and reopen the browser, the cart items that have not been removed should still appear on the page.

Bonus A: Update cart item quantities and reset values

Instead of displaying text values of the item quantities (e.g. “1”, “5”, “12”) on the page, replace those with an HTML5 number spinner. What’s that, you ask? It’s really just a fancy way to say an input field that lets you increment and decrement a number value. We’ve actually already used one on the page. Check the input for `quantity` for an example.

When a user increments or decrements the input value of an item’s quantity, get the input value and use it to update the corresponding key-value pair in `localStorage`.

Bonus B: Calculate the item totals

Currently, if a user adds an item to the cart, let’s say **Apples: 1**, and then adds that same item to the cart again with a different value, let’s say**Apples: 3**, the value in `localStorage` gets overwritten with the new value -- **Apples: 3**.

Write a function to calculate the total quantity added to the cart, instead of rewriting the value. Using the above example, the actual total would be**Apples: 4**. After calculating the total quantity of an item, update the corresponding key-value pair in `localStorage` with the total.

When you’ve finished all of the above, congratulate yourself for all the hard work. You made a shopping cart!