



JavaScript Web Storage – Lab 19

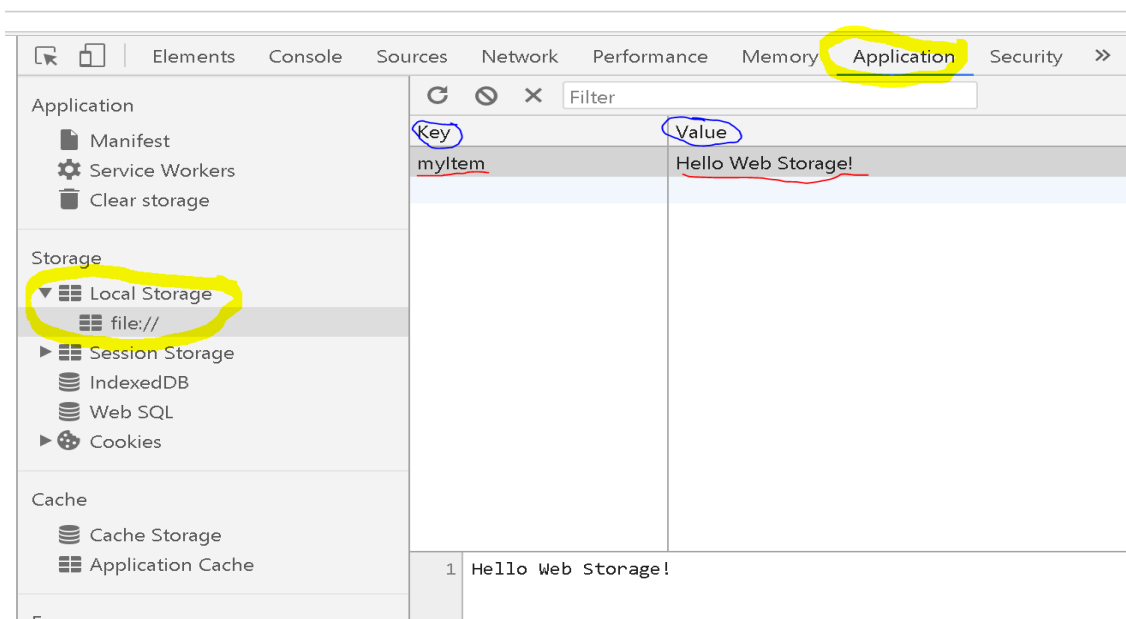
This week we will look at web storage in JavaScript. Create a new folder for this lab and save each step in the exercises as a separate file.

Exercise 1 – Save an item to local storage.

Create a basic web page and include the following code in your JavaScript:

```
localStorage.myItem = "Hello Web Storage!";
```

Save and run your page. This line of code above will create a storage item in your browser called “myItem” and it will save the text “Hello Web Storage” with it. To confirm this, right-click anywhere inside this page, and select ‘inspect’ (or inspect element). Once the developer tools window is open in your browser, select the “application” tab, then “Local Storage” and then “file”, as shown below:



You should see the key and value that you entered in your code. Once complete, close this page.

Exercise 2 – Retrieve an item from local storage.

1. In this exercise you will create a second web page that will retrieve the item that you saved in exercise 1.

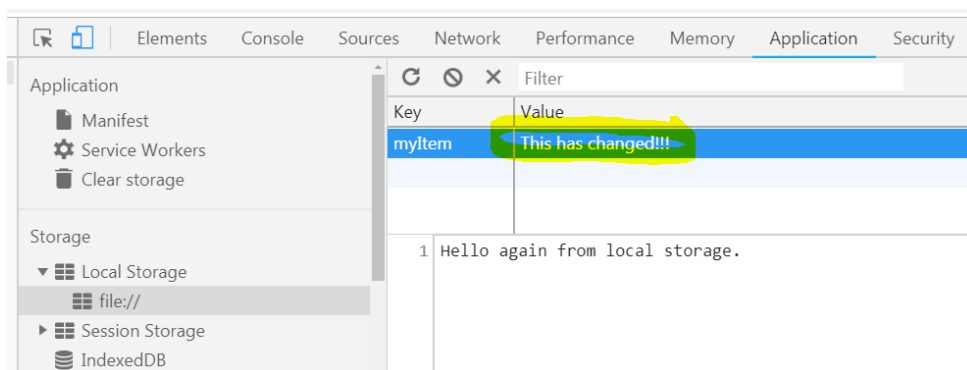
Create a second web page that includes the following code in your JavaScript:

```
let myVar = localStorage.myItem;  
alert(myVar);
```

This code creates a variable called myVar and retrieves the data in localStorage (with a key of “myItem”) and places this into your variable. When the page runs, you should see a pop up box that displays the information that you stored in the first exercise.

Close and reopen your browser and check if the local storage still contains your item.

2. Change the value stored in localStorage. Open your first page, and change the value stored to “Hello again from local storage”. Save your page and run it. Open your second page and run it again. Your alert should reflect the change you made.
3. Change the value stored in localStorage using the developer tools. Open developer tools, and navigate to local storage. Change the value by double-clicking on it, as shown below:



Open your second page and run it again. Your alert should reflect the change you made.

Exercise 3 – Save and retrieve user input to local storage.

1. Open both pages you created in the previous exercises. You will be working with these pages for a while, so for clarity, add a `<h1>` heading on each page stating which page it is – eg., “This is Page 1” or “This is Page 2”. You should also create a link on each page (using the `<a>` tag) that links page 1 to page 2, and vice versa, as shown below.



On page 1, add an input box and button that will allow the user to type information in the input box and save to local storage by click the button.



You will need to create these elements in HTML, and then create a function in JavaScript that collects this information. For example, your code could be similar to below:

```
function save() {  
    localStorage.myNewItem = document.getElementById("inbox").value;  
}
```

This code above will be activated when the user clicks the “Save to local storage” button you created. The function will need to use a DOM query to gather whatever data was typed in by the user. Note that the DOM query in the example above is specifying an element that has an ID of “inbox”. If you are using this ID in your code, then make sure to give your HTML input element the same ID.

Test your page out by running it and then opening the Web Developer tools to confirm that the item has been created in local web storage.

2. On page 2, add a button in your HTML. When this button is clicked, it should retrieve the item from part 1 of this exercise and display it on the screen, as shown below:



Close and reopen your browser on your second page and check if the local storage still contains your item.

Exercise 4 – Clear storage

Add a button that will empty the local storage, as shown below.



The screenshot shows a web application interface. At the top, it says "This is page 1" in a large, bold, black serif font. Below this, there is a purple link that says "Go to page 2". At the bottom, there is a form with the label "Enter Name:" followed by a text input field. To the right of the input field are two buttons: "Save to local storage" and "Clear Local Storage".

Confirm that local storage is cleared by checking the application tab in the inspector, under local storage.

Exercise 5 – Session Storage

Repeat exercises 1 to 4 using session storage. Session storage will only save the data while the “session” active. Once you close the browser, session storage items will be lost. When repeating the exercises, close the browser at end of each exercise to see the differences between local storage and session storage.

Session storage uses the same syntax as local storage, for example:

```
sessionStorage.ssessStore = "This is some session storage data";
```

Exercise 6 – Basic Shopping Cart

Create a basic 3 page website site. The first page is a **HOME** page, that asks the user to input their name, and stores their name to local storage. Your page can be similar to as shown below:

[HOME](#) [SHOP](#) [CART](#)

Enter Your name:

The second page is a **SHOP** page, allowing the user to select quantities for an item as shown below. The shopping cart item quantity should be saved to local storage.

Note: Image for this exercises is available in images.zip on Moodle. Just download and unzip this file to access the image.

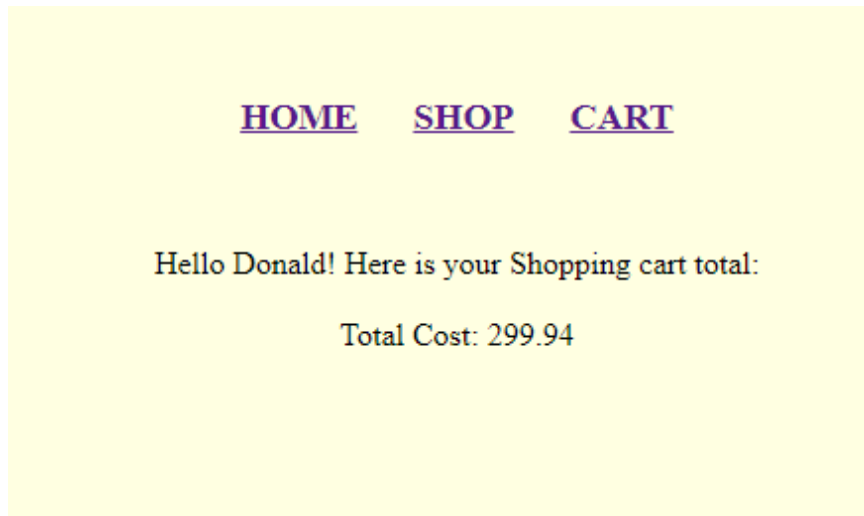
[HOME](#) [SHOP](#) [CART](#)



Amazon echo Dot
Price: €49.99

Quantity:

The third page or **CART** page should display a greeting to the user, along with the contents of the cart and a calculation of the total cost, as shown here:



Exercise 7 – Shopping Cart with Multiple items

Amend your code from exercise 6 to include an additional 2 items on 2 new pages. The shop page should include links to 3 pages – one for ordering each product. The cart page should display the total price based on the item price and quantity for each product ordered.

There should also be an option to empty the cart.

Images for suggested items are available in images.zip which is included with this lab on Moodle, but you can use images you like if you prefer!

Items descriptions you can use are as follows:

Amazon echo Dot @ €49.99




Beats Solo Wireless On-Ear Headphones @ €231

Samsung Gear Smart Watch @ €340

Exercise 8 – Shopping Cart with Multiple items

Using the example from the lecture, which is available on Moodle along with this lab, [lab19.zip](#), amend this example to include one additional product. You will need to update the home.html page to include an extra product and add images to this page:

home.html

Home	Cart	Checkout
 <p>Socks cost €5 a pair.</p> <p>How many pairs of socks do you wish to order?</p> <p>Qty: <input type="text"/></p> <p><input type="button" value="Add to cart"/></p>	 <p>Shoes cost €100 a pair.</p> <p>How many pairs of shoes do you wish to order?</p> <p>Qty: <input type="text"/></p> <p><input type="button" value="Add to cart"/></p>	 <p>Runners cost €50 a pair.</p> <p>How many pairs of runners do you wish to order?</p> <p>Qty: <input type="text"/></p> <p><input type="button" value="Add to cart"/></p>

Note: Images are available in the images.zip file on Moodle on along with this lab.

The cart.html page will need to be updated to include the extra product:

cart.html

[Home](#)
[Cart](#)
[Checkout](#)

#	Product	Cost per pair	Quantity ordered	Total(€)
1	Socks	5	1	5
2	Shoes	100	1	100
2	Runners	50	1	50
			Total Cost(€)	155

An additional page is also required, for checkout. You will need to add links for this page to the other pages. See previous screenshots. This page will include a form.

checkout.html

Home	Cart	Checkout
----------------------	----------------------	--------------------------

Name:

Address:

Email Address:

Promo Code:

☐ Fast Delivery (Extra €3):

Total Cost:

In checkout.html, you need to include the following JavaScript functionality.

- Add a blur event to the promo code field so that when this field loses focus, a check is done to see if the code entered by the user is “footwear”. If it is, then apply a 10% discount to the total cost, and update the Total Cost field with the updated total.
- Add a click event to the Free Delivery checkbox. If this box is checked, then update add €3 to the total cost and update the Total Cost field with the updated total. If the user then unchecks the checkbox, subtract the €3 from the total cost, and update the Total Cost field with the updated total.
- On clicking the submit button perform the following validation:
 - Check that name and address fields are not blank. If they are blank, display an alert indicating this.
 - Check a valid email address was entered – that it contains an @ symbol and a dot(.). If the email address doesn't contain these, display an alert indicating this. You can use the includes() function for this task.