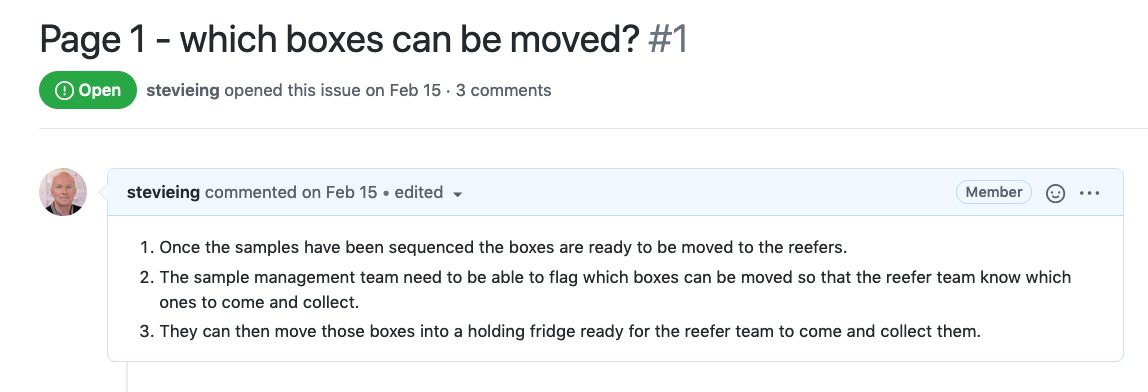
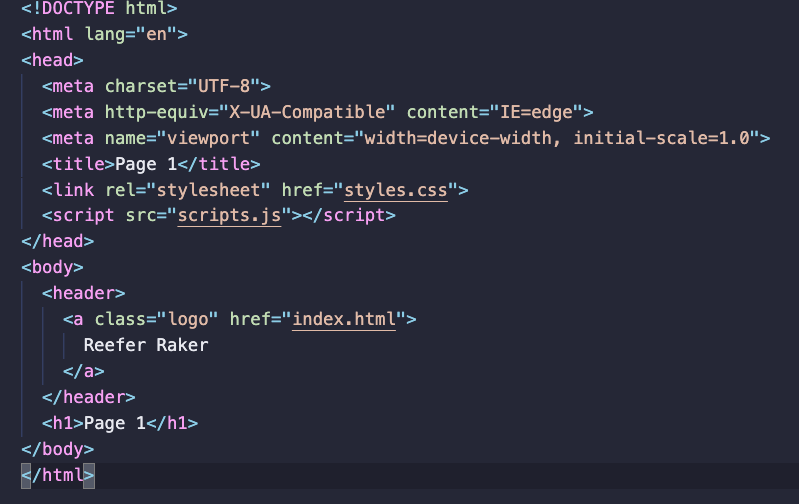
# Reefer Raker workshop – Thursday April 1 2pm

Today we are going to work on a prototype for the reefer raker.

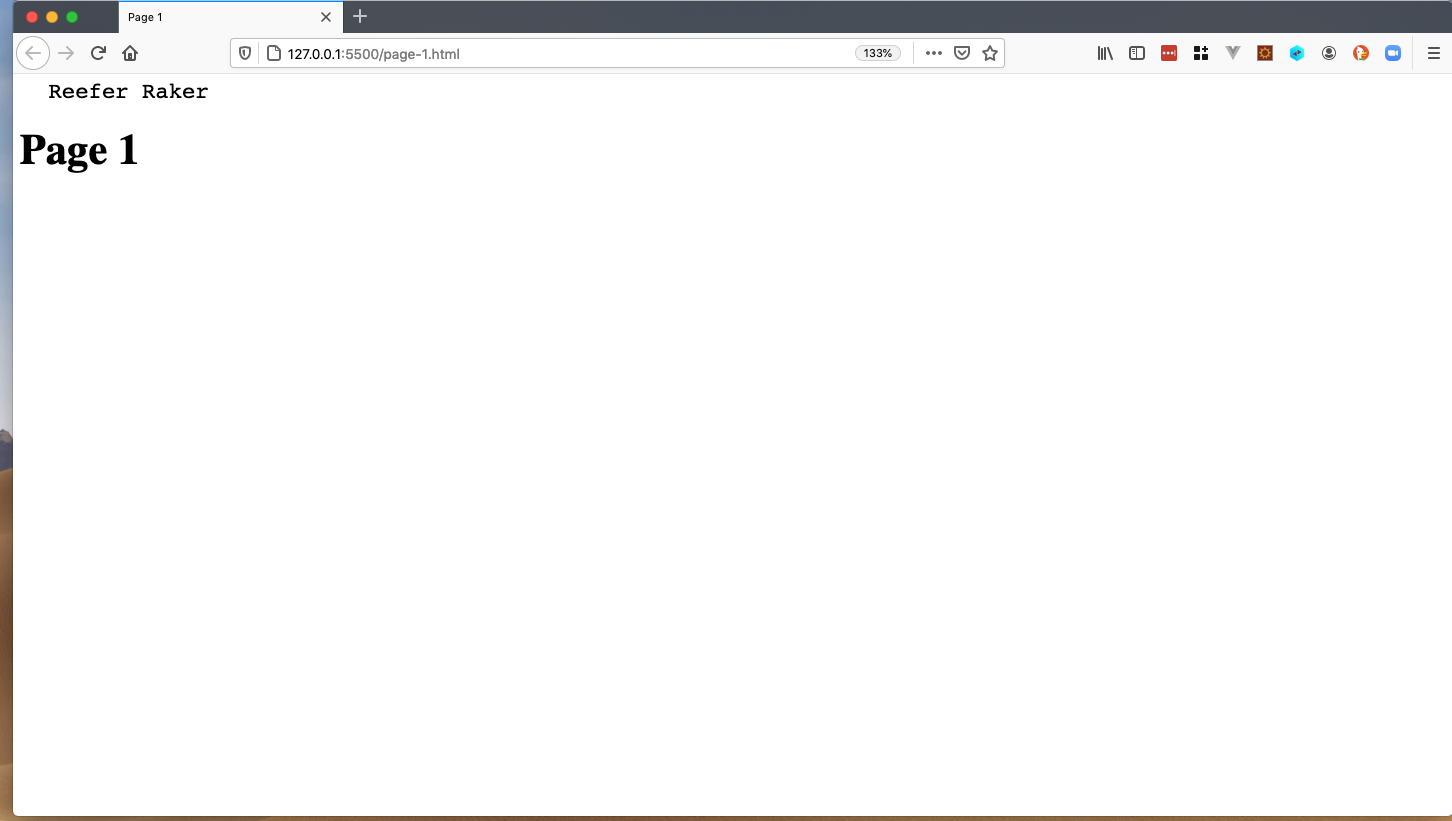
And we are going to concentrate on the first issue – page 1 which boxes can be moved?



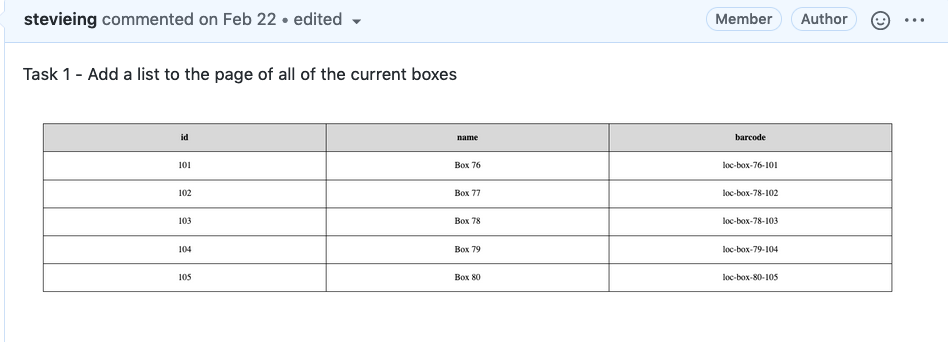
1. First thing to do – read the issue carefully.
2. Before you start you will need to checkout the Github repository and open it in your text editor and then open the blank page 1.



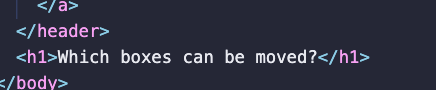
1. For the purpose of this exercise you will need to check how it looks so open a browser and show page 1 and you should see a blank page.



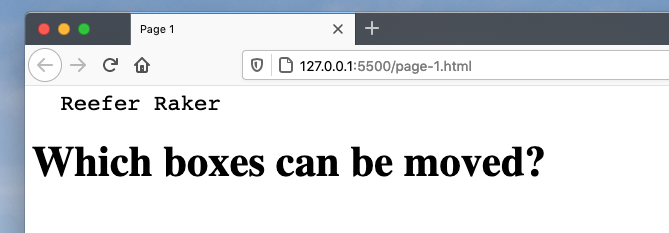
1. We need to look at the instructions for task 1. We are going to create a table that looks like the picture. Simple!



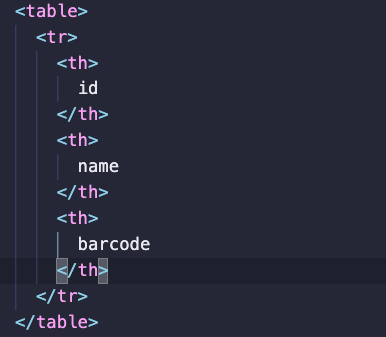
1. Add a title (h1) ‘which boxes can be moved?’



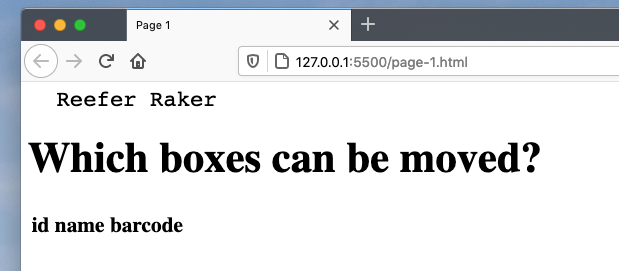
1. Open in browser. And voila we have written our first bit of code!



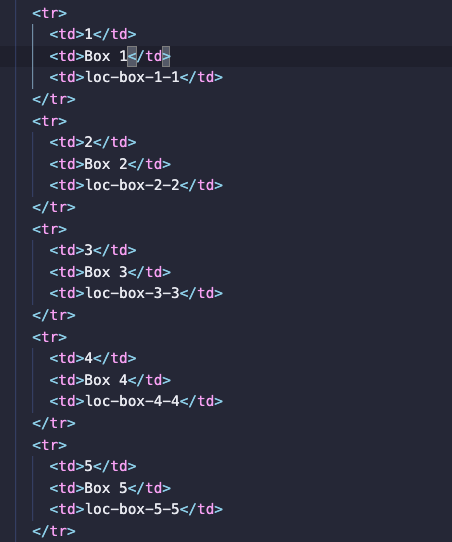
1. Next thing to do is create a table and within that we will create a header row, with each of the headers id, name and barcode.



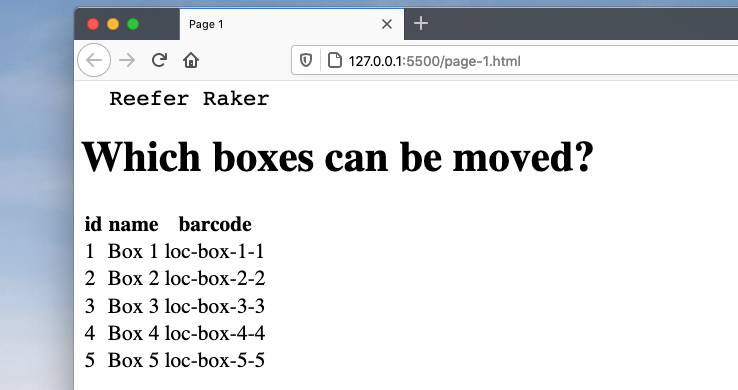
1. And again, Open in browser. You should see the headings



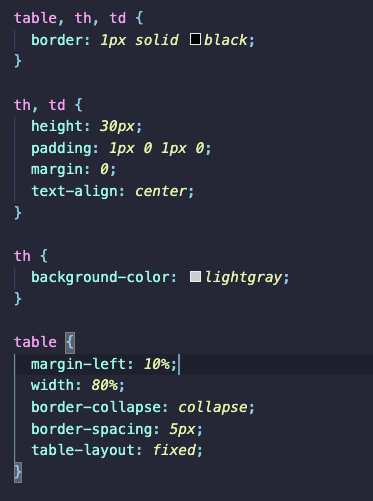
1. Now add some rows with data. You can make the data up yourself. This is a prototype so it doesn’t have to be real. I will add 5 rows each with 3 columns with an id, name and barcode.



1. Open a browser to check what you have done. You should see 5 rows with some data.

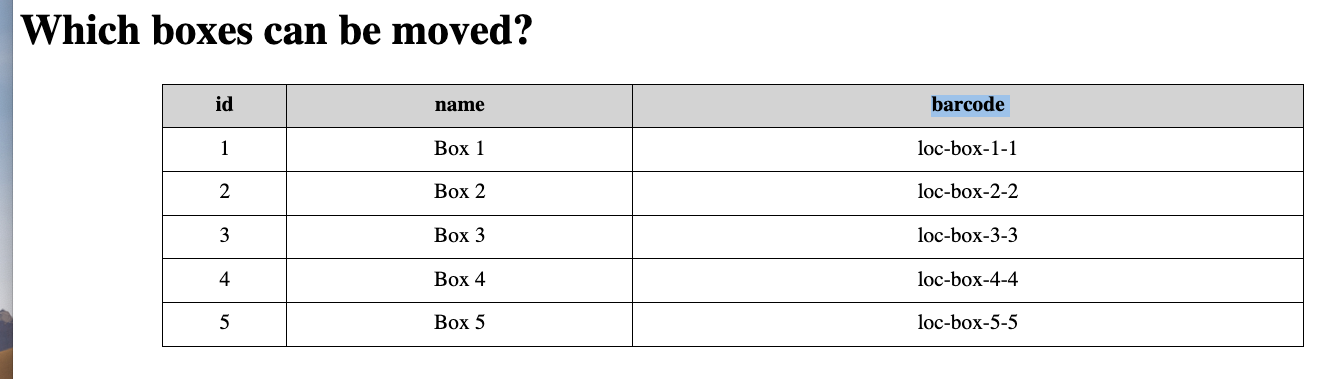


1. As it stands the data is all there but it looks a bit rubbish so we need to add some styling to make it look like a proper table. Hop across to the styles.css page and add the following …

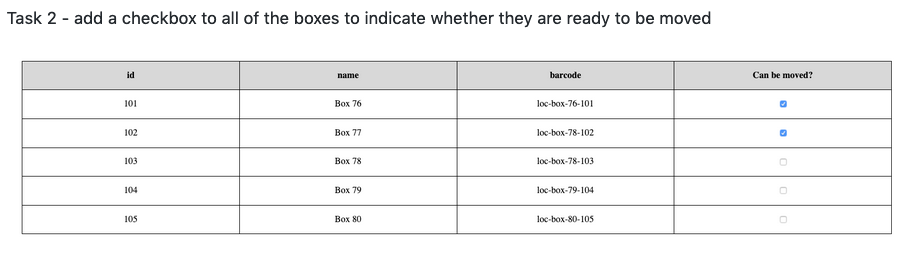


* Add a border to the table and the cells.
* Fix the height add a little bit of padding and set the margin to 0.
* Make sure the text is centred.
* Add a grey border to the header to make it stand out.
* For the table itself we can add a margin to the left.
* Make the width of the table to 80% so it will be 80% of the page in the middle.
* We can also set border-collapse to collapse. This means that all of the separate borders will be collapsed into a single border which looks much better.
* Add a spacing around the border

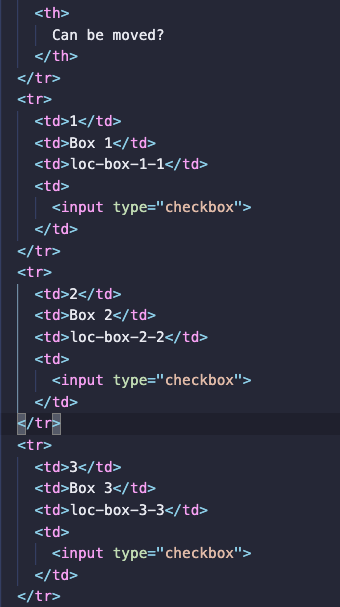
1. And open in browser. Now we have a table that looks ok. Not a design classic but it is structured, tidy and makes sense.



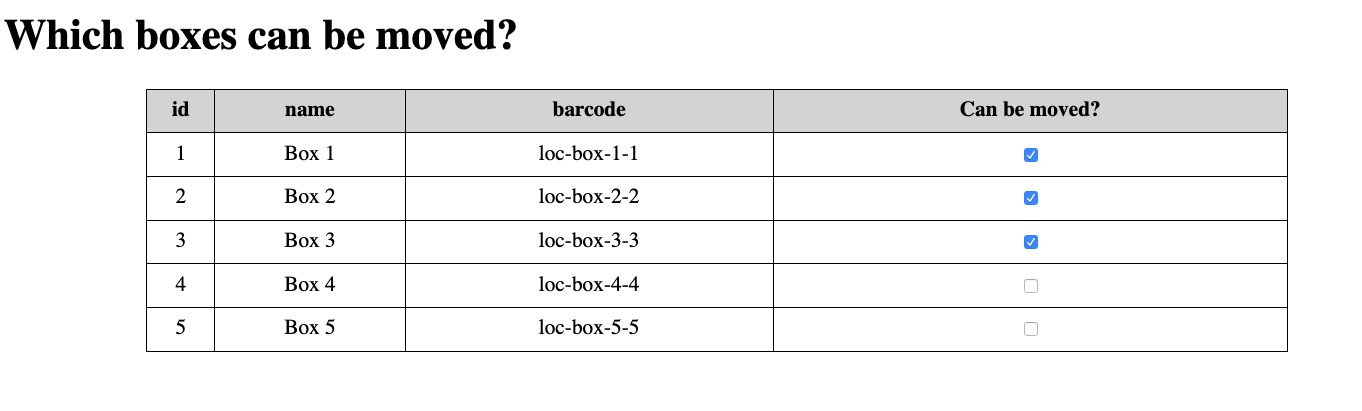
1. So now on to task 2. Again, read the instructions carefully.



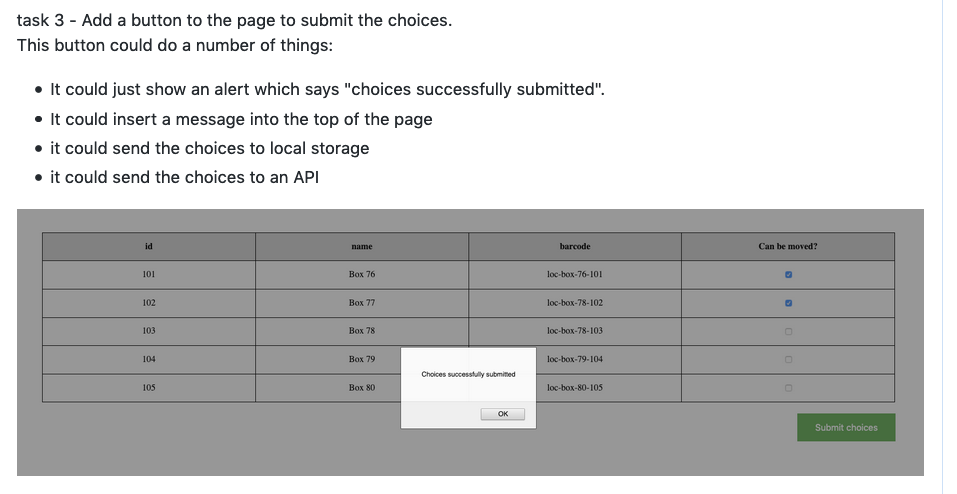
1. Add a new header column titled ‘can be moved?’ and then add a checkbox for each row.



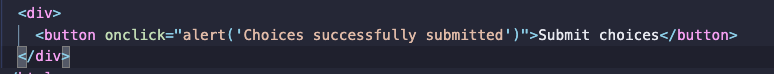
1. Open it in a browser and check a few of the boxes just to make sure it works.



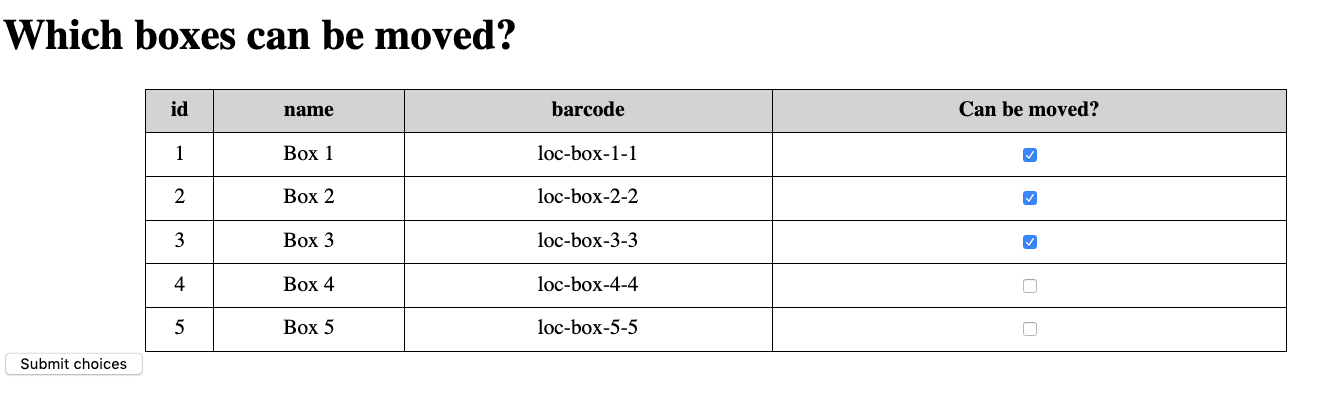
1. So on to Task 3 …. This one is a bit trickier, so again make sure you read the instructions carefully.



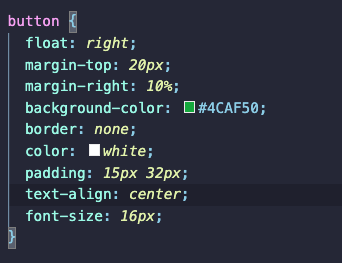
1. There are several different options but I am going to firstly just show an alert but then I am going to add some JavaScript to save the data to local storage.
2. Firstly, add a div after the table. Within the div add a button with the text ‘Submit choices’ and then add an onclick event with an alert which will show ‘Choices successfully submitted.



1. Open in a browser and press the button and you should see a pop-up window saying ‘choices successfully submitted’. But what you will notice is that the button looks basic and it is not very prominent so we are going to jazz it up a little bit.

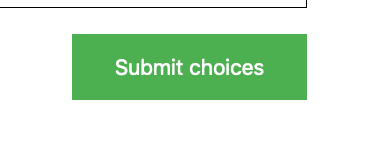


1. So, pop over to your styles.css and add the following

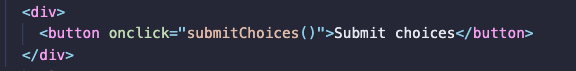


* If we float the button to the right it moves the button to the right of the page and so makes it more prominent.
* If we add a margin to the top and to the right it separates it from the table and aligns it.
* If we add a background colour of green, align the text, make it bigger and make the text white it means the button looks so much better.
* We can also remove any hint of a border and add some padding so the text has some space around it.

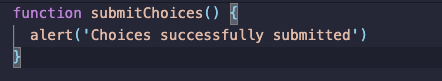
1. Open a browser window and your button should look so much better.



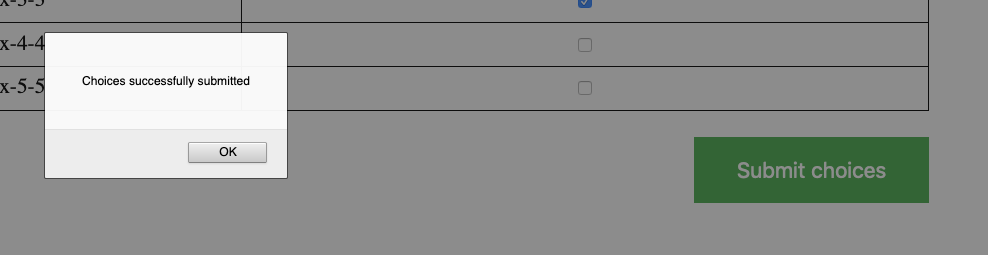
1. Now for the tricky bit. We are going to add some JavaScript so we can submit the data to local storage. First change the onclick event to a function submitChoices.



1. And then add a function in the scripts.js file called submitChoices. Add an alert within the function which says ‘choices submitted’.



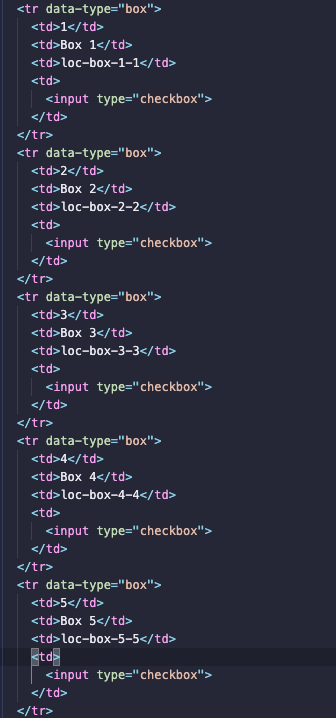
1. Open in browser and you should see that pressing the button does the same as it did before but now we are using our own function.



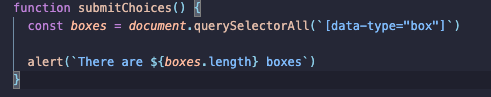
1. So now what we need to do is extract all of the data from the page, put it into a suitable format and save it in local storage. So how do we do that …
2. First thing we need to do is identify the data in the page and we do that using data tags.

* The <data> tag is used to add a machine-readable translation of a given content.
* This element provides both a machine-readable value for data processors, and a human-readable value for rendering in a browser.
* This data tag will be used by our function to identify all of the rows which have boxes.

1. We are going to add a data type with the value box to each of the rows like so

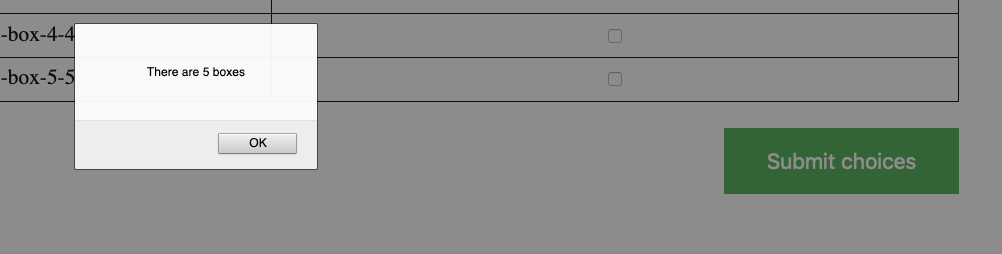


1. Then you can pop across to the scripts file and add some code to pick up those data-type tags. Create a variable called boxes and we will use a query selector to pick up those tags and show an alert on the page to show us how many boxes there are.



* We are using the HTML DOM Document Object
* The document object represents your web page in a variable
* If you want to access any element in an HTML page, you always start with accessing the document object.
* In our case we are using the querySelectorAll function to get all of the tags which contain the data type box. This returns an array of objects and we can use the length property to find out how many there are.

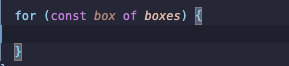
1. And if you open it in a browser and press the submit choices button as you can see it shows the correct number of boxes. It worked. Result. We are nearly there.



1. What we need to do now is extract the data for each box and put that into a variable. Like we did with the data type tag we need to add the data attribute tag for the id, name and barcode for each box so we can extract that information like so …



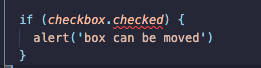
1. So now we can pop across to the scripts file so we can add some functionality to extract the data. First add a for loop for each box. This for loop will loop through all of the boxes we created earlier and create a variable for each box object.



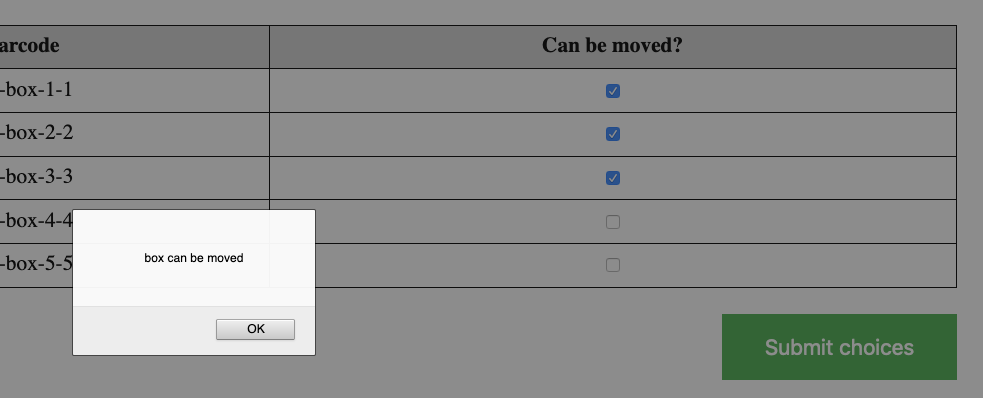
1. Next create a variable and use a query selector to get the checkbox object.



1. Remember we only need to extract the data if the box is ready to be moved so we need to check if the checkbox has actually been checked. To do that we can add an if statement like so and use the checked property which will equate to true if the user has checked the box. And to check that this is actually working we can add an alert for each box that has been checked.



1. Open in browser check some boxes, press the button and hopefully you should see the correct number of ‘box can be moved’ messages. Hopefully you should see the same number of boxes appear as the number of boxes that have been checked.



1. Next, we can extract the data for each box and add it to a variable.

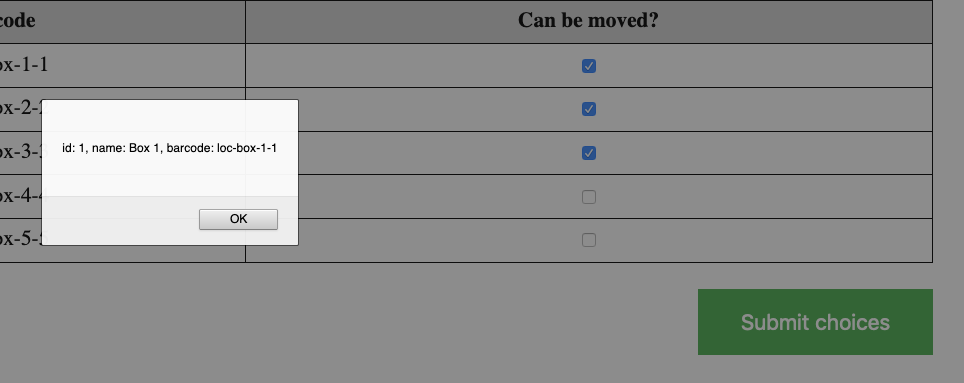
* Create a variable for the item as an object
* Create the id property from the query selector inner html for the data attribute for id
* Create the name property from the query selector inner html for the data attribute for name
* Create the barcode property from the query selector inner html for the data attribute for barcode



1. Again, we need to make sure that this is working so we can add an alert for each box to check that the correct data is extracted like so …



1. And then we can check this is working by (you guessed it) opening the browser!



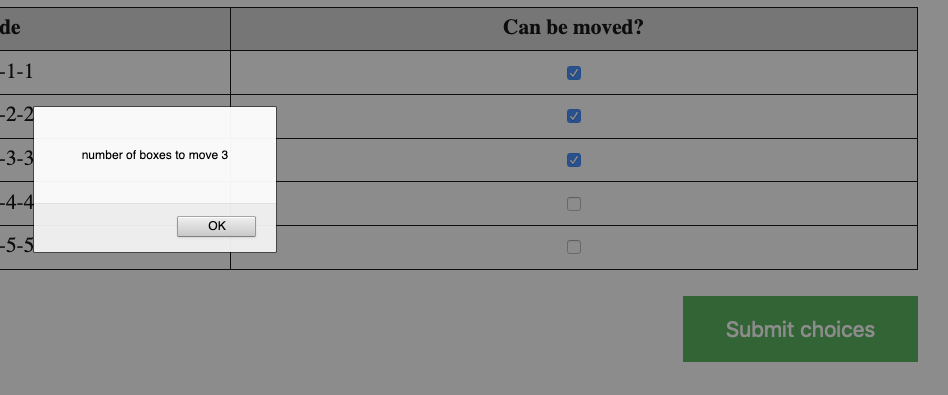
1. Next, we need to create a variable which will be an array of all of the objects that we have created and we want to push each object onto that array.



1. And again, we want to check that it is working by showing an alert of the number of boxes that can be moved.



1. Then if we open that in a browser we should see how many boxes can be moved.



1. Finally, we want to add this to localStorage. which is HTML web storage

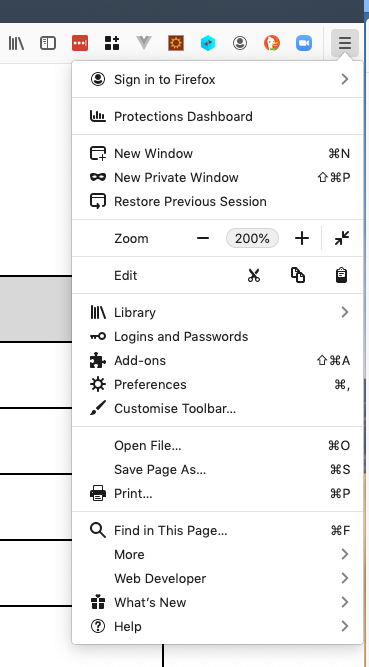
## What is HTML Web Storage?

* With web storage, web applications can store data locally within the user's browser.
* Before HTML5, application data had to be stored in cookies, included in every server request. Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance.
* Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.
* Local storage is amazing and allows you to store stuff in your browser. It is still there when your browser closes.

1. One of local storages limitations is that it only allows you to store strings so we need to turn our object into a string using the JSON.stringify function and then create an item in local storage called boxes like so!



1. So that’s it apart from we need to make sure it is working so back to the browser and check some of the boxes.
2. In your browser of choice, you need to be able to see the local storage. I am using Firefox so on the menu at the right-hand side there is a little box with 3 lines and when you click on that there should be a menu item web developer.



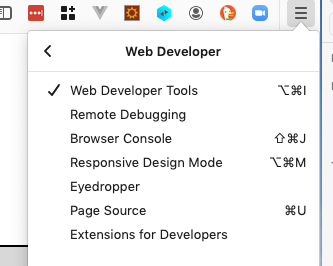
1. Unfortunately, I am not going to be able to show this if you are using Microsoft IE or edge or Chrome but the ways to get to it are similar. My advice would be to do a search for how to access the developer tools for those two browsers.

* IE - The first step is to launch the app in Internet Explorer and open up the **Developer Tools**. You can do this with the keyboard using the F12 key or by selecting “F12 **Developer Tools**” in the “**Tools**” menu. The **Developer Tools** will now be open inside the browser tab, and the DOM Explorer tab will be active.
* **In IE11, you can see local storage in console on dev tools:**
* Show dev **tools** (press F12 )
* Click "Console" or press Ctrl + 2.
* Type **localStorage** and press Enter.

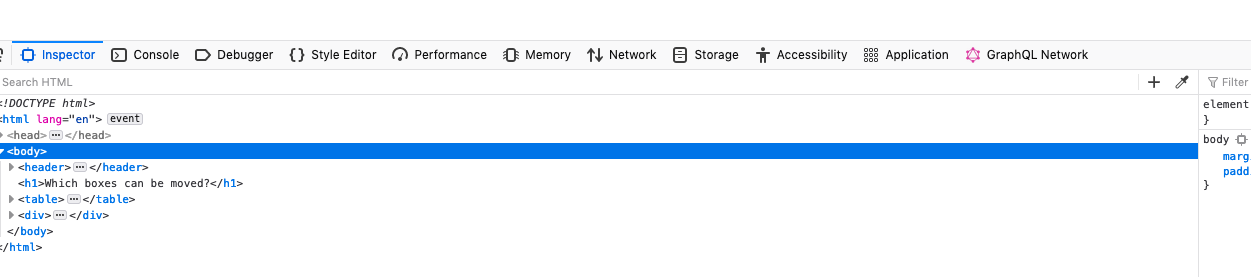
## Open DevTools from Chrome's main menu

* Click **Customize and control Google Chrome** Customize and control Google Chromeand then select **More Tools** > **Developer Tools**. For chrome there should be a storage option.

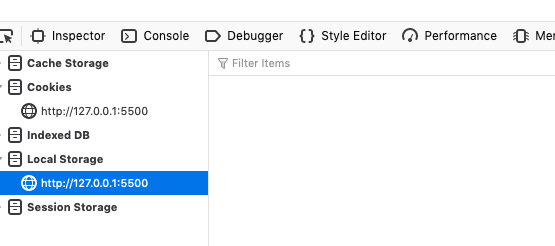
1. Click on the web developer menu item and select web developer tools.



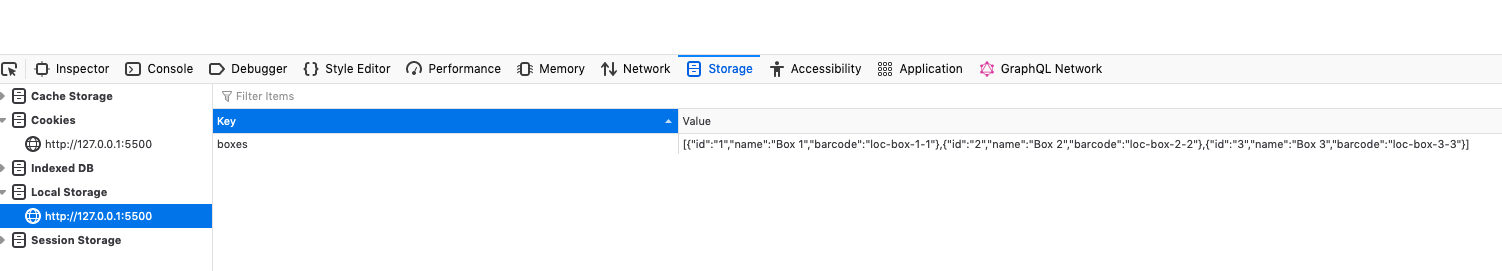
1. You should now have a new view in the bottom of your screen.



1. And then you need to select the Storage option and then highlight local storage which should be empty or at least not contain a boxes item.



1. So now if you click the button the data should appear in your local storage hopefully!



Wrap up …

Hopefully this has given you a bit of insight into how I do web development. This is just a snapshot of how I would do things but the take home from this is:

* Take small, slow steps
* Test what you have done as often as possible
* You might think it takes more time but actually from bitter experience I have spent much more time fixing problems when I don’t do it this way.

Now it would be great if you could go away and practise this and try it again with the next issue. It is more difficult but if you tackle it in the same way you should make steady progress.

It would be great to see how you get on so please use the discussions on github or maybe even make a pull request. I would be more than happy to take a look.

Any questions …