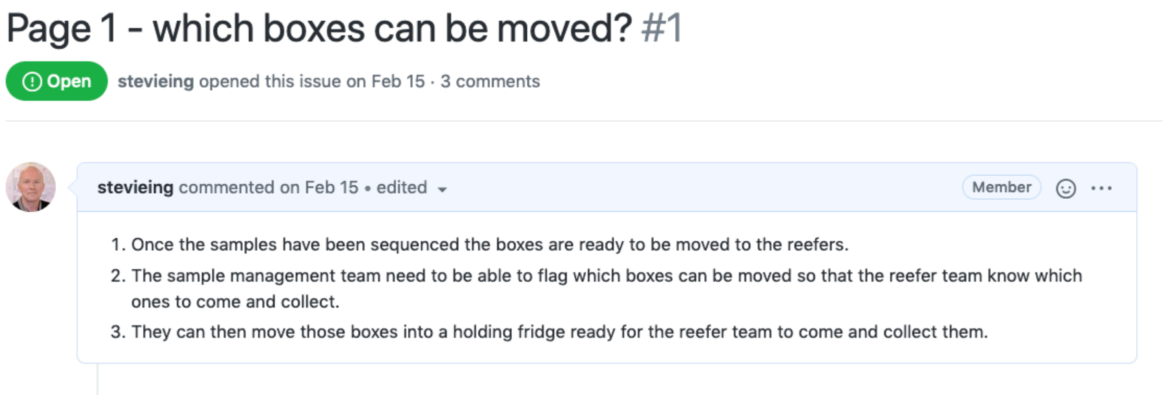
Reefer Raker workshop – Thursday April 1 2pm

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?



Page 1 - which boxes can be moved? #1

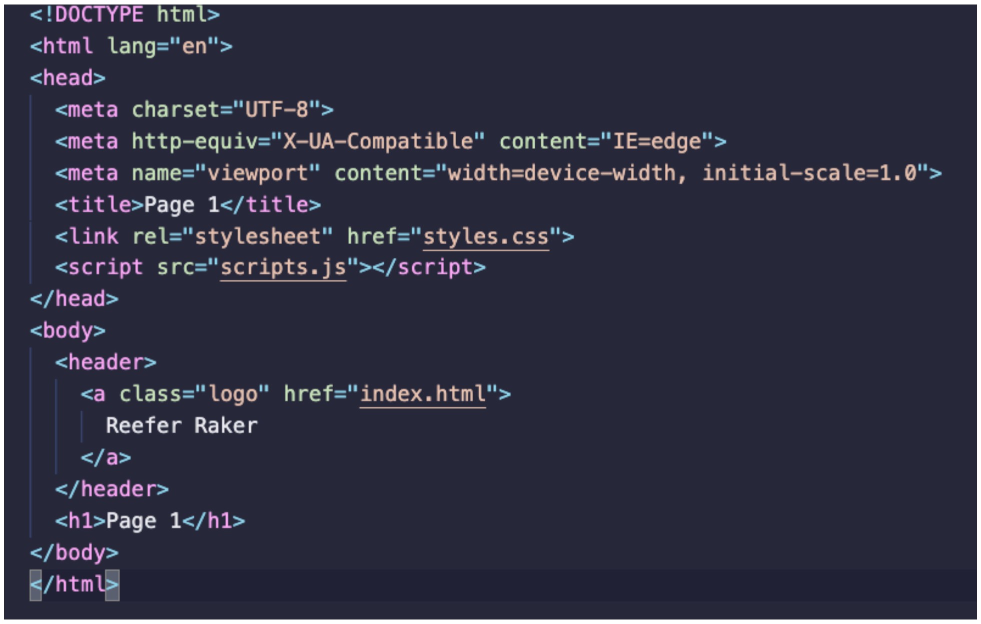
Open stevieing opened this issue on Feb 15 · 3 comments

stevieing commented on Feb 15 • edited

1. Once the samples have been sequenced the boxes are ready to be moved to the reefers.
2. The sample management team need to be able to flag which boxes can be moved so that the reefer team know which ones to come and collect.
3. They can then move those boxes into a holding fridge ready for the reefer team to come and collect them.

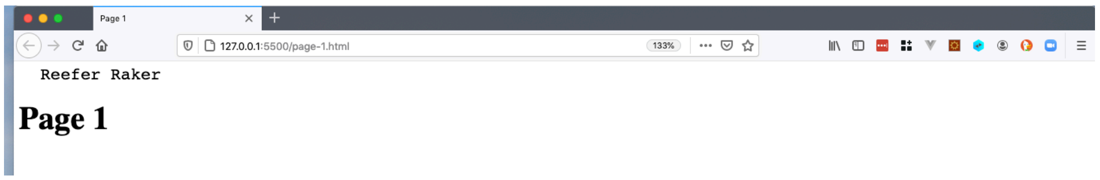
1. First thing to do – read the issue carefully.

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.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Page 1</title>
  <link rel="stylesheet" href="styles.css">
  <script src="scripts.js"></script>
</head>
<body>
  <header>
    <a class="logo" href="index.html">
      Reefer Raker
    </a>
  </header>
  <h1>Page 1</h1>
</body>
</html>
```

3. 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.

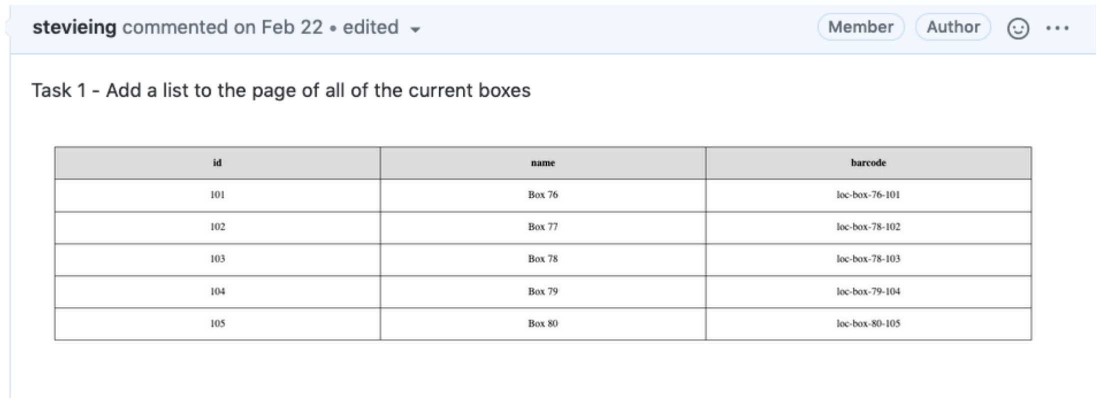


127.0.0.1:5500/page-1.html

Reefer Raker

Page 1

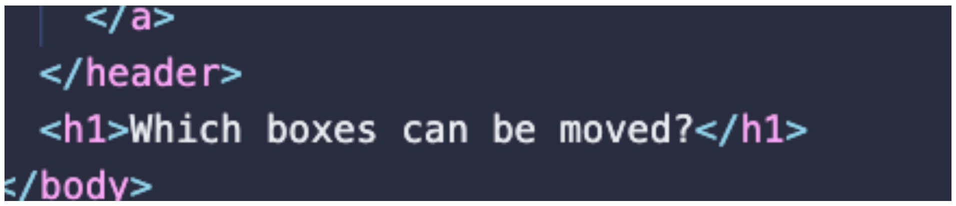
4. We need to look at the instructions for task 1. We are going to create a table that looks like the picture. Simple!



Task 1 - Add a list to the page of all of the current boxes

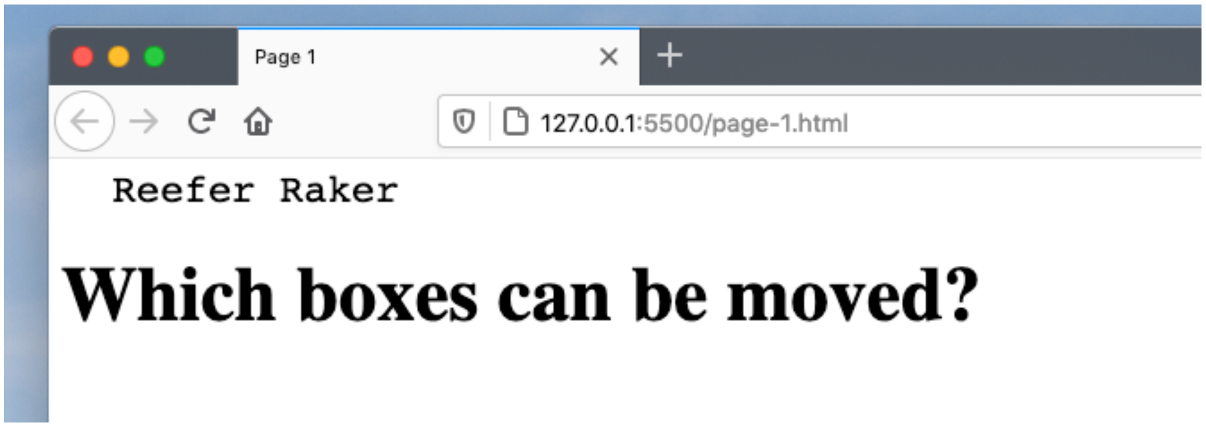
id	name	barcode
101	Box 76	loc-box-76-101
102	Box 77	loc-box-78-102
103	Box 78	loc-box-78-103
104	Box 79	loc-box-79-104
105	Box 80	loc-box-80-105

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



```
    </a>
  </header>
  <h1>Which boxes can be moved?</h1>
</body>
```

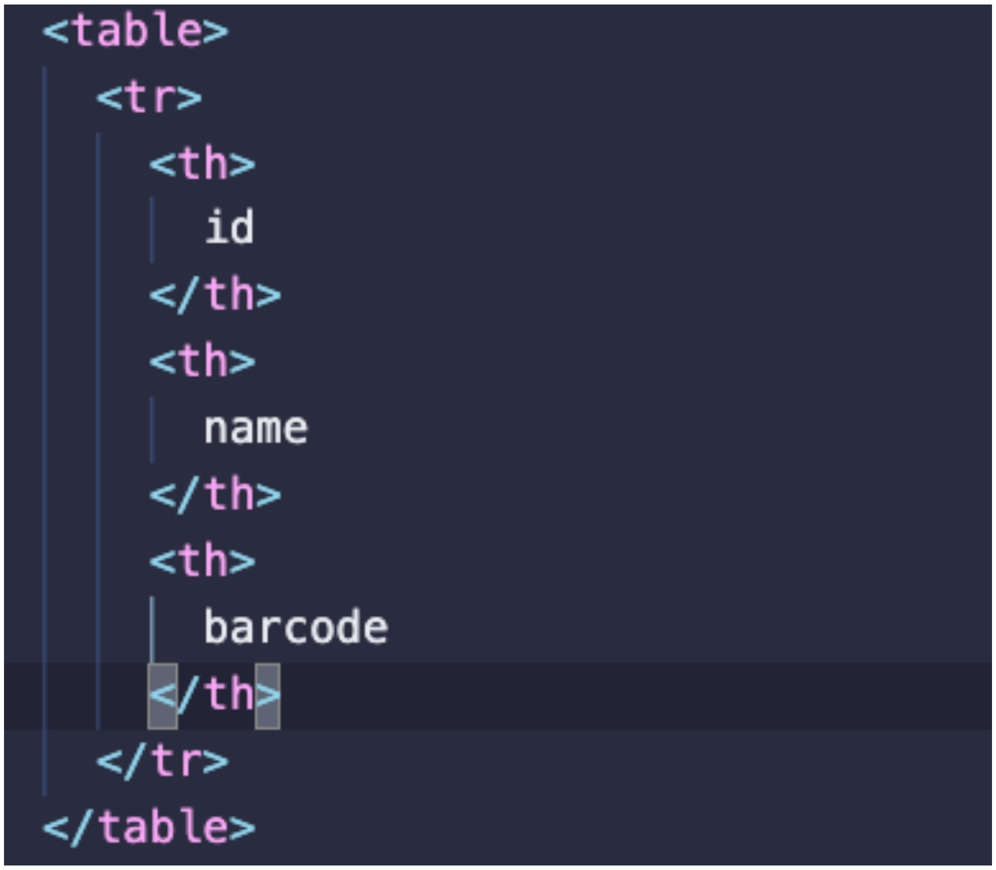
6. Open in browser. And voila we have written our first bit of code!



Reefer Raker

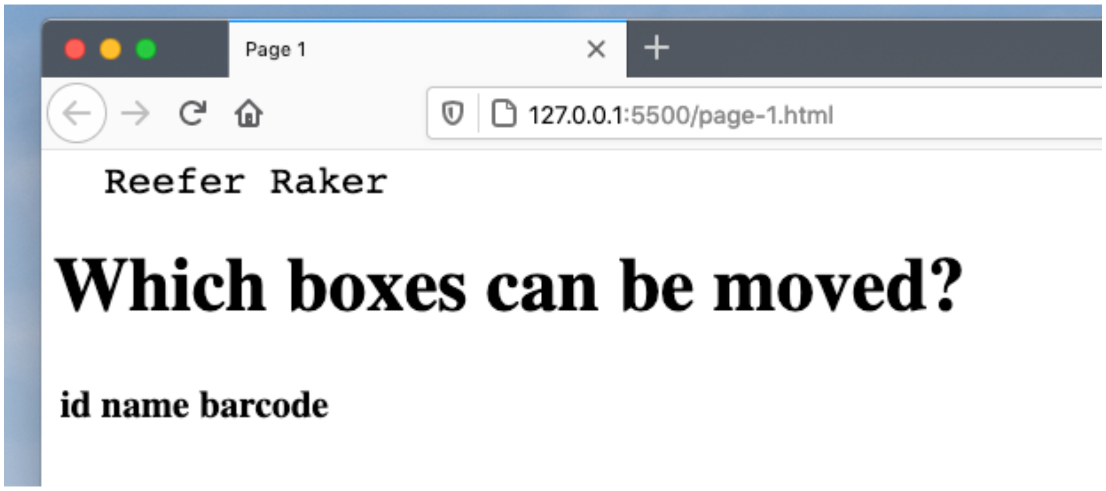
Which boxes can be moved?

7. 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.



```
<table>
  <tr>
    <th>
      id
    </th>
    <th>
      name
    </th>
    <th>
      barcode
    </th>
  </tr>
</table>
```

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

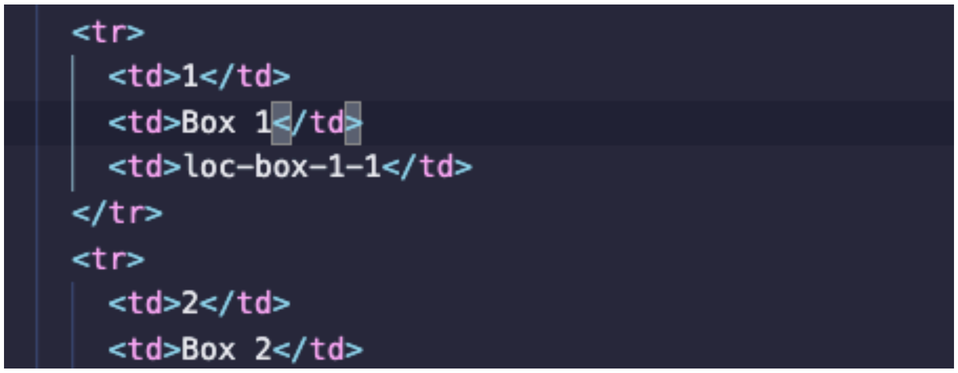


Reefer Raker

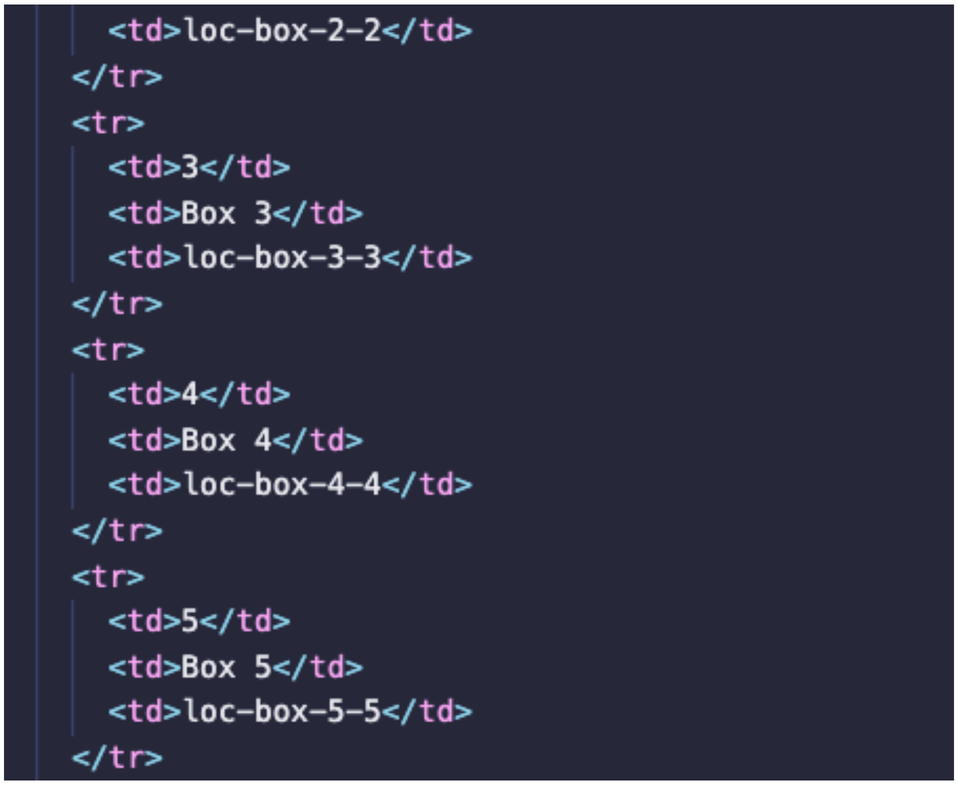
Which boxes can be moved?

id name barcode

9. 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.

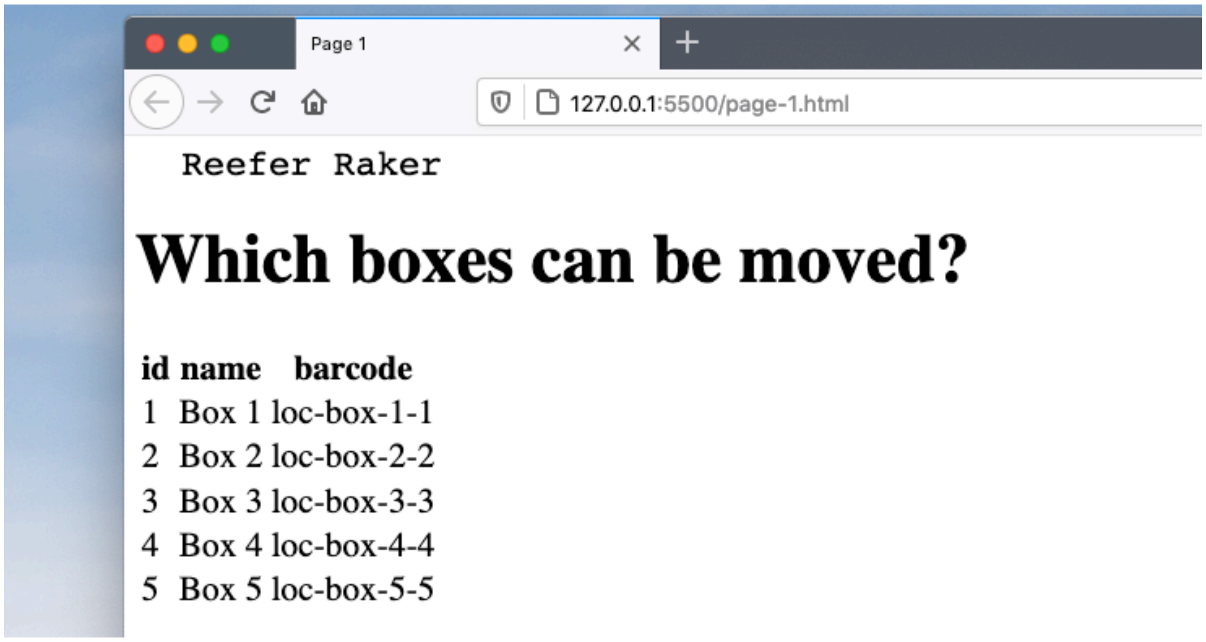


```
<tr>
  <td>1</td>
  <td>Box 1</td>
  <td>loc-box-1-1</td>
</tr>
<tr>
  <td>2</td>
  <td>Box 2</td>
```



```
    <td>loc-box-2-2</td>
  </tr>
  <tr>
    <td>3</td>
    <td>Box 3</td>
    <td>loc-box-3-3</td>
  </tr>
  <tr>
    <td>4</td>
    <td>Box 4</td>
    <td>loc-box-4-4</td>
  </tr>
  <tr>
    <td>5</td>
    <td>Box 5</td>
    <td>loc-box-5-5</td>
  </tr>
```

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

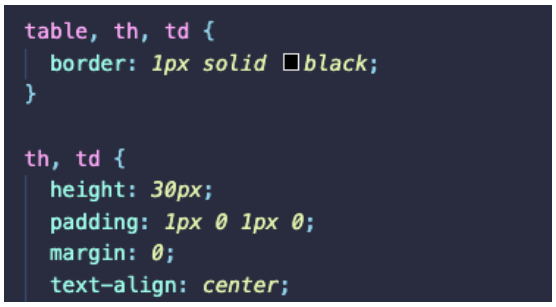


Reefer Raker

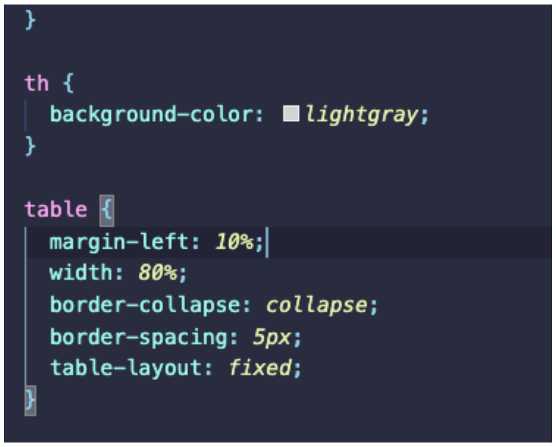
Which boxes can be moved?

id	name	barcode
1	Box 1	loc-box-1-1
2	Box 2	loc-box-2-2
3	Box 3	loc-box-3-3
4	Box 4	loc-box-4-4
5	Box 5	loc-box-5-5

11. 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 …



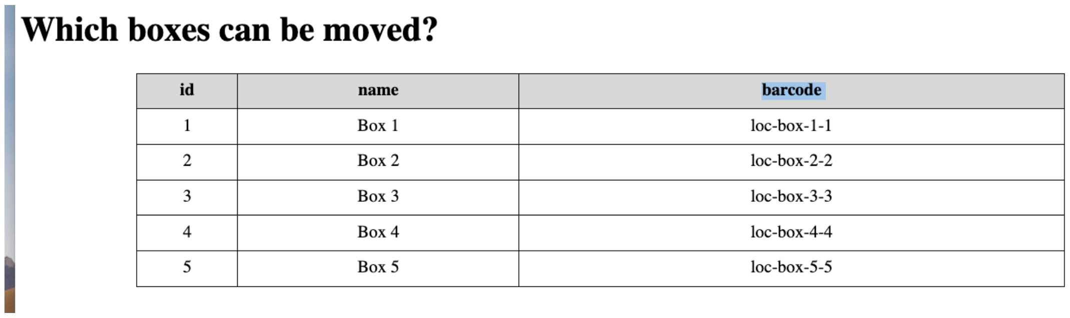
```
table, th, td {
  border: 1px solid black;
}

th, td {
  height: 30px;
  padding: 1px 0 1px 0;
  margin: 0;
  text-align: center;
```



```
}

th {
    background-color: lightgray;
}

table {
    margin-left: 10%;
    width: 80%;
    border-collapse: collapse;
    border-spacing: 5px;
    table-layout: fixed;
}
```

- 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

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

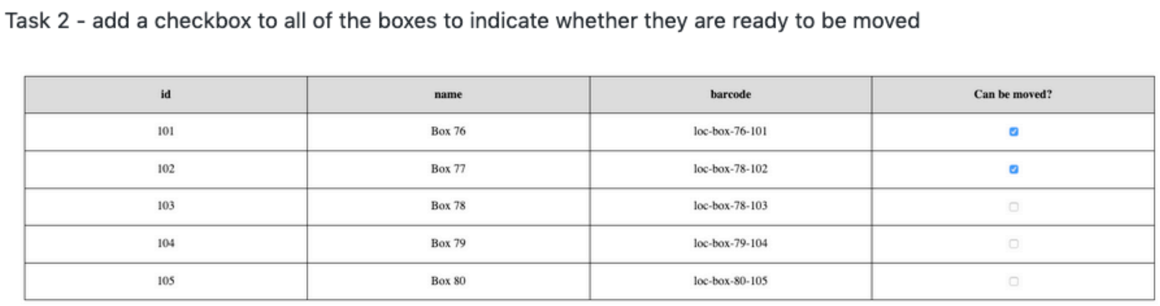


Which boxes can be moved?

id	name	barcode
1	Box 1	loc-box-1-1
2	Box 2	loc-box-2-2
3	Box 3	loc-box-3-3
4	Box 4	loc-box-4-4
5	Box 5	loc-box-5-5

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

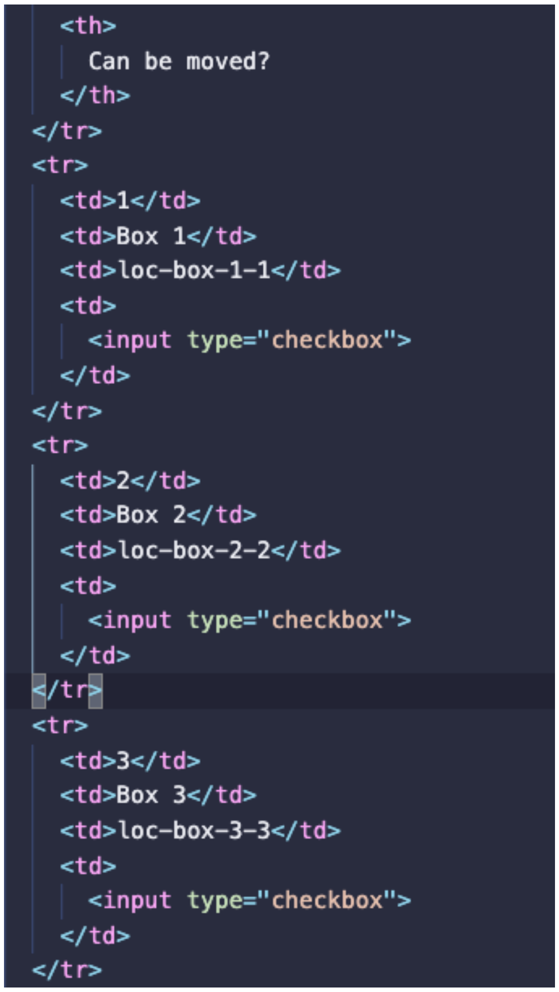
Using an ide is useful in things like this because it can highlight syntax errors.



Task 2 - add a checkbox to all of the boxes to indicate whether they are ready to be moved

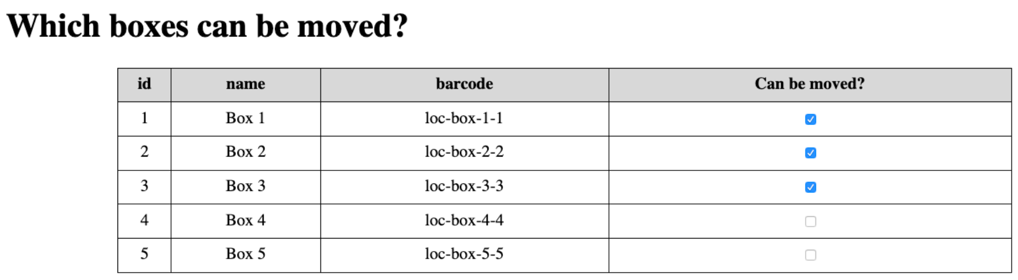
id	name	barcode	Can be moved?
101	Box 76	loc-box-76-101	☑
102	Box 77	loc-box-78-102	☑
103	Box 78	loc-box-78-103	☐
104	Box 79	loc-box-79-104	☐
105	Box 80	loc-box-80-105	☐

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



```
    <th>
      Can be moved?
    </th>
  </tr>
  <tr>
    <td>1</td>
    <td>Box 1</td>
    <td>loc-box-1-1</td>
    <td>
      <input type="checkbox">
    </td>
  </tr>
  <tr>
    <td>2</td>
    <td>Box 2</td>
    <td>loc-box-2-2</td>
    <td>
      <input type="checkbox">
    </td>
  </tr>
  <tr>
    <td>3</td>
    <td>Box 3</td>
    <td>loc-box-3-3</td>
    <td>
      <input type="checkbox">
    </td>
  </tr>
```

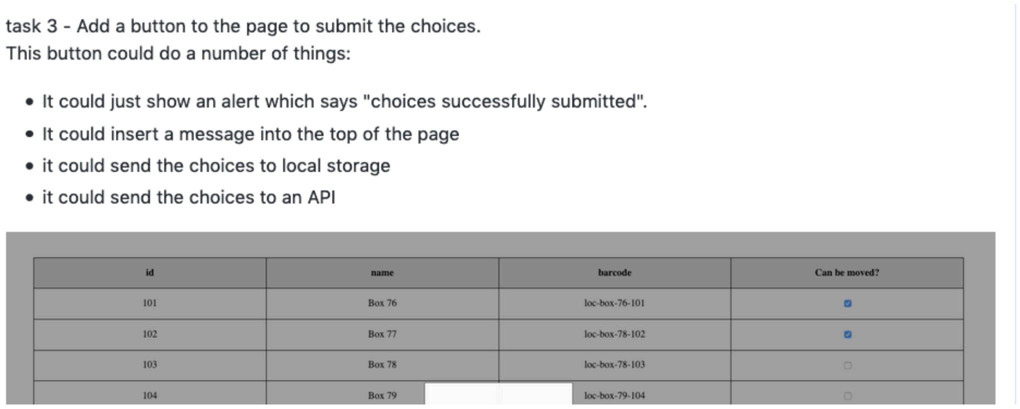
15. Open it in a browser and check a few of the boxes just to make sure it works.



Which boxes can be moved?

id	name	barcode	Can be moved?
1	Box 1	loc-box-1-1	☑
2	Box 2	loc-box-2-2	☑
3	Box 3	loc-box-3-3	☑
4	Box 4	loc-box-4-4	☐
5	Box 5	loc-box-5-5	☐

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



task 3 - Add a button to the page to submit the choices.
This button could do a number of things:

- It could just show an alert which says "choices successfully submitted".
- It could insert a message into the top of the page
- it could send the choices to local storage
- it could send the choices to an API

id	name	barcode	Can be moved?
101	Box 76	loc-box-76-101	☑
102	Box 77	loc-box-78-102	☑
103	Box 78	loc-box-78-103	☐
104	Box 79	loc-box-79-104	☐

Choices successfully submitted

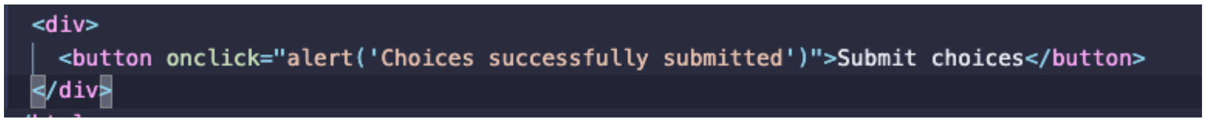
OK

105	Box 80	loc-box-80-105	▢

Submit choices

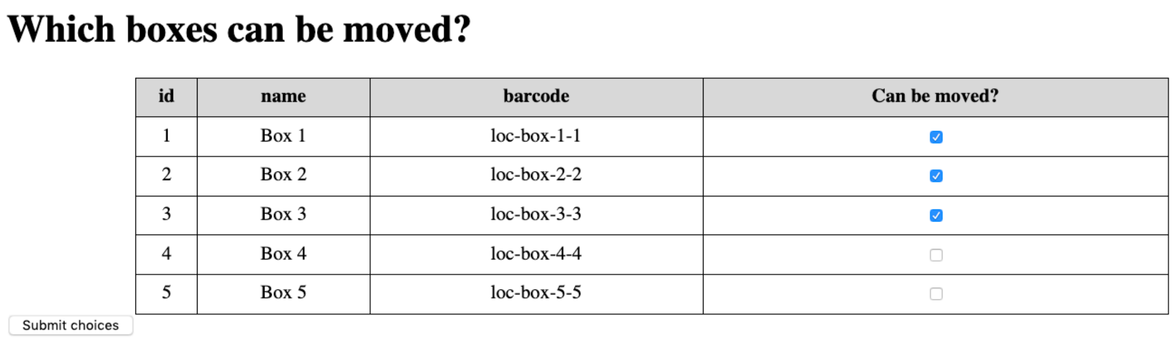
17. There are several different options but I am going to firstly just show an alert but then I am going to add some JavaScript to save the data to local storage.

18. 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.



```
<div>
  <button onclick="alert('Choices successfully submitted')">Submit choices</button>
</div>
```

19. 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.

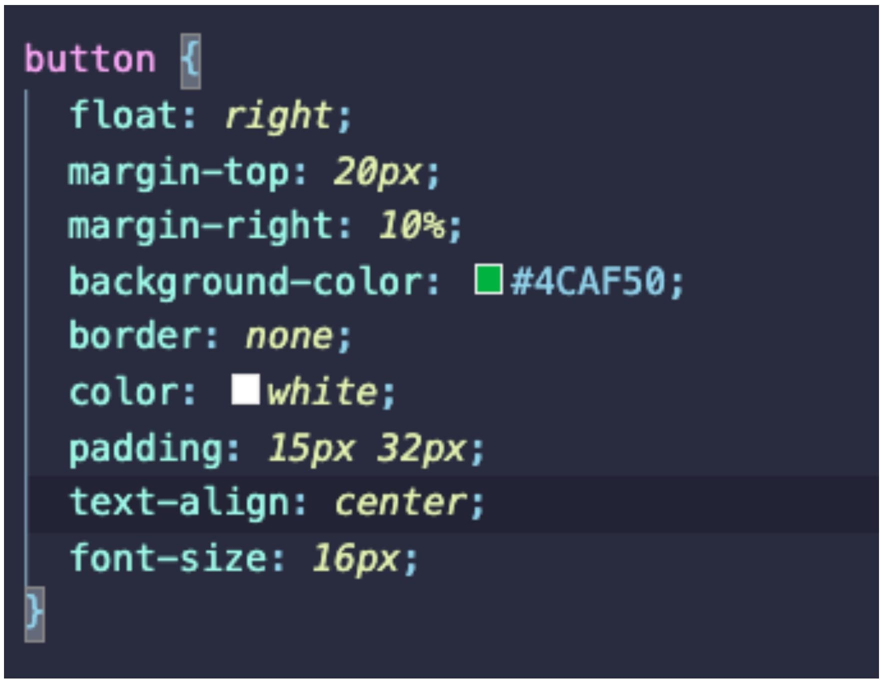


Which boxes can be moved?

id	name	barcode	Can be moved?
1	Box 1	loc-box-1-1	✓
2	Box 2	loc-box-2-2	✓
3	Box 3	loc-box-3-3	✓
4	Box 4	loc-box-4-4	▢
5	Box 5	loc-box-5-5	▢

Submit choices

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

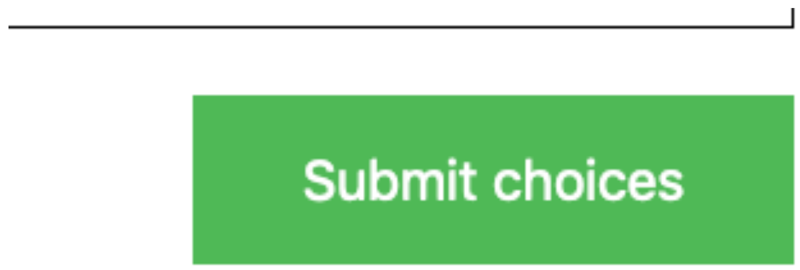


```
button {
  float: right;
  margin-top: 20px;
  margin-right: 10%;
  background-color: #4CAF50;
  border: none;
  color: white;
  padding: 15px 32px;
  text-align: center;
  font-size: 16px;
}
```

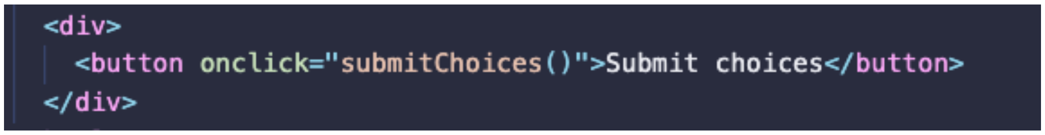
• If we float the button to the right it moves the button to the right of the page and so makes it

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.

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

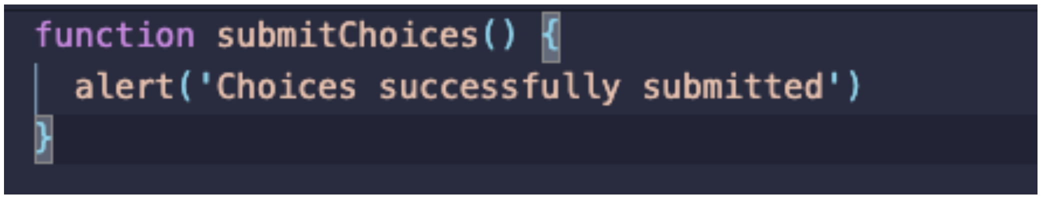


22. Now for the tricky bit. We are going to add some JavaScript so we can submit the data to local storage. First change the onclick event to a function submitChoices.



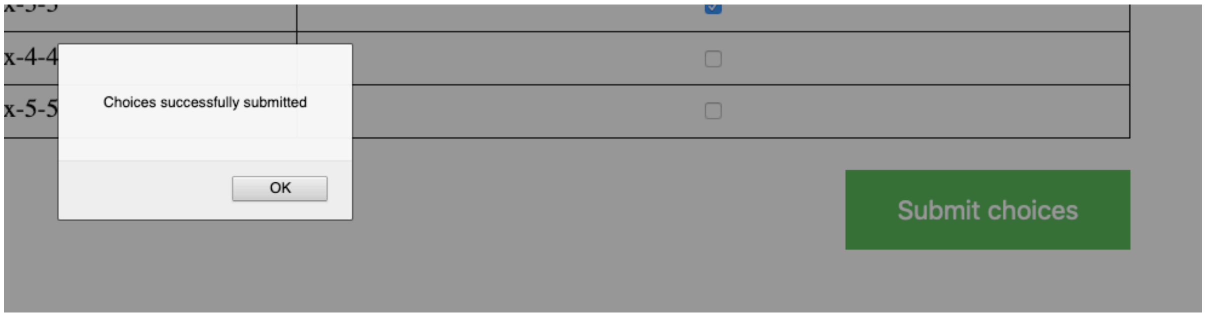
```
<div>
  <button onclick="submitChoices()">Submit choices</button>
</div>
```

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



```
function submitChoices() {
  alert('Choices successfully submitted')
}
```

24. 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.



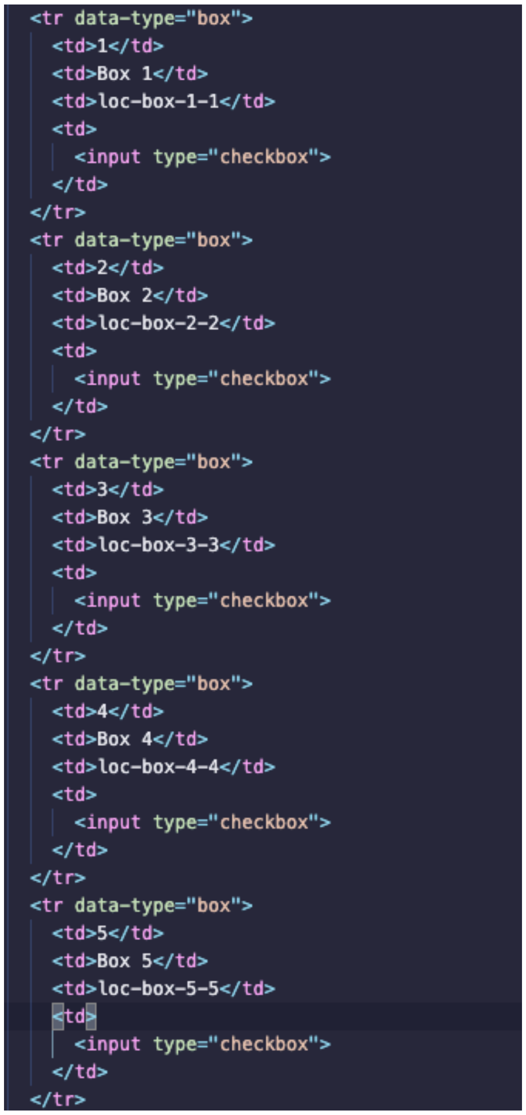
Choices successfully submitted

25. 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 …

26. First thing we need to do is identify the data in the page and we do that using data tags.

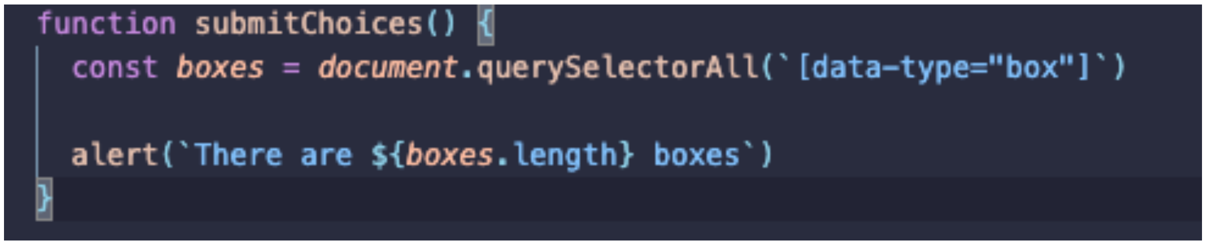
- The `<data>` tag is used to add a machine-readable translation of a given content.
- This element provides both a machine-readable value for data processors, and a human-readable value for rendering in a browser.
- This data tag will be used by our function to identify all of the rows which have boxes.

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



```html
<tr data-type="box">
  <td>1</td>
  <td>Box 1</td>
  <td>loc-box-1-1</td>
  <td>
    <input type="checkbox">
  </td>
</tr>
<tr data-type="box">
  <td>2</td>
  <td>Box 2</td>
  <td>loc-box-2-2</td>
  <td>
    <input type="checkbox">
  </td>
</tr>
<tr data-type="box">
  <td>3</td>
  <td>Box 3</td>
  <td>loc-box-3-3</td>
  <td>
    <input type="checkbox">
  </td>
</tr>
<tr data-type="box">
  <td>4</td>
  <td>Box 4</td>
  <td>loc-box-4-4</td>
  <td>
    <input type="checkbox">
  </td>
</tr>
<tr data-type="box">
  <td>5</td>
  <td>Box 5</td>
  <td>loc-box-5-5</td>
  <td>
    <input type="checkbox">
  </td>
</tr>
```

28. 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.



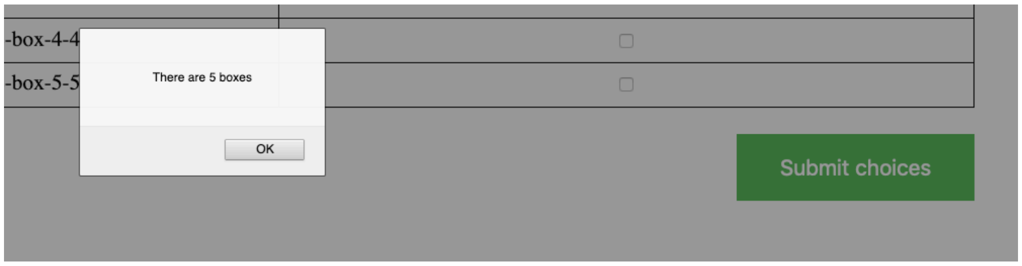
```javascript
function submitChoices() {
  const boxes = document.querySelectorAll(`[data-type="box"]`)

  alert(`There are ${boxes.length} boxes`)
}
```

- We are using the HTML DOM Document Object
- The document object represents your web page in a variable

The document object represents your web page in a variable

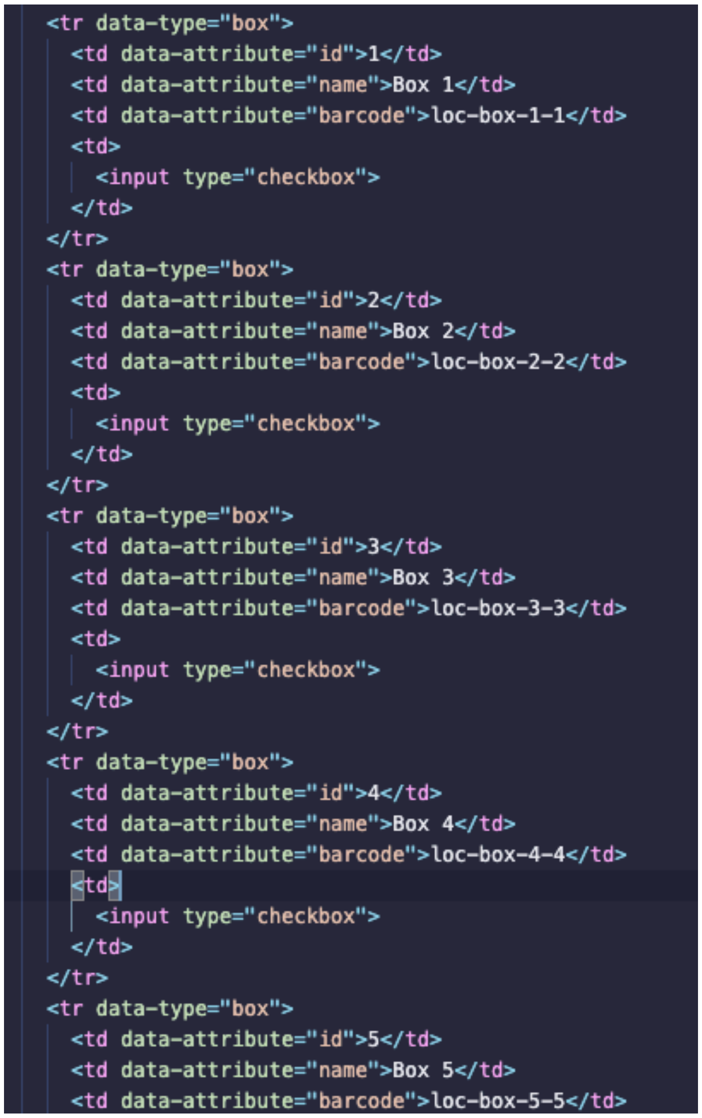
- If you want to access any element in an HTML page, you always start with accessing the document object.
- In our case we are using the querySelectorAll function to get all of the tags which contain the data type box. This returns an array of objects and we can use the length property to find out how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

29. 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.

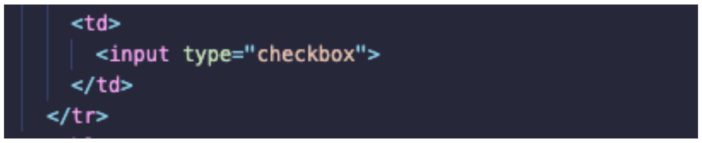


There are 5 boxes

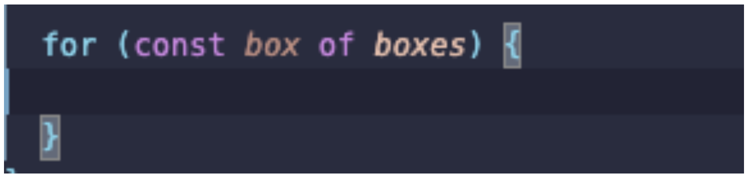
30. What we need to do now is extract the data for each box and put that into a variable. Like we did with the data type tag we need to add the data attribute tag for the id, name and barcode for each box so we can extract that information like so …



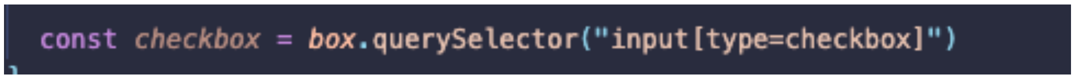
```
<tr data-type="box">
  <td data-attribute="id">1</td>
  <td data-attribute="name">Box 1</td>
  <td data-attribute="barcode">loc-box-1-1</td>
  <td>
    <input type="checkbox">
  </td>
</tr>
<tr data-type="box">
  <td data-attribute="id">2</td>
  <td data-attribute="name">Box 2</td>
  <td data-attribute="barcode">loc-box-2-2</td>
  <td>
    <input type="checkbox">
  </td>
</tr>
<tr data-type="box">
  <td data-attribute="id">3</td>
  <td data-attribute="name">Box 3</td>
  <td data-attribute="barcode">loc-box-3-3</td>
  <td>
    <input type="checkbox">
  </td>
</tr>
<tr data-type="box">
  <td data-attribute="id">4</td>
  <td data-attribute="name">Box 4</td>
  <td data-attribute="barcode">loc-box-4-4</td>
  <td>
    <input type="checkbox">
  </td>
</tr>
<tr data-type="box">
  <td data-attribute="id">5</td>
  <td data-attribute="name">Box 5</td>
  <td data-attribute="barcode">loc-box-5-5</td>
```


```
    <td>
      <input type="checkbox">
    </td>
  </tr>
```

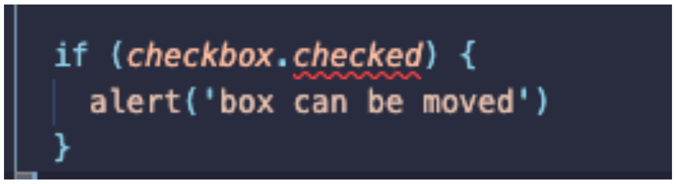
31. So now we can pop across to the scripts file so we can add some functionality to extract the data. First add a for loop for each box. This for loop will loop through all of the boxes we created earlier and create a variable for each box object.


```
for (const box of boxes) {

}
```

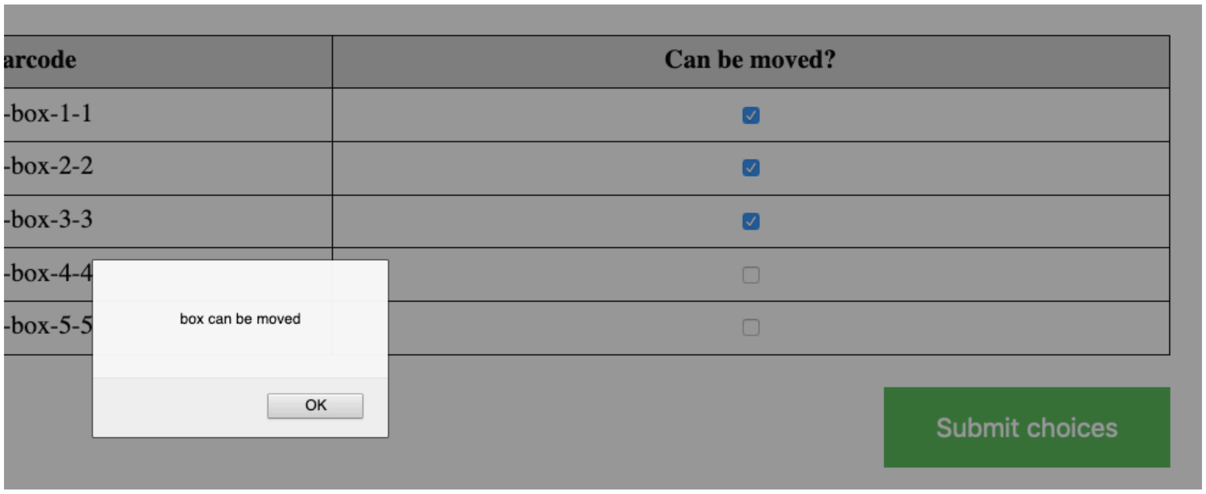
32. Next create a variable and use a query selector to get the checkbox object.


```
const checkbox = box.querySelector("input[type=checkbox]")
```

33. 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.


```
if (checkbox.checked) {
  alert('box can be moved')
}
```

34. 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.



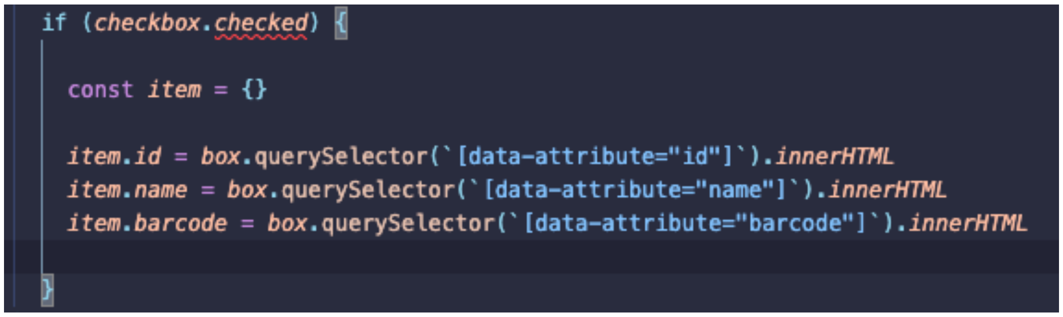
arcode	Can be moved?
-box-1-1	✓
-box-2-2	✓
-box-3-3	✓
-box-4-4	☐
-box-5-5	☐

box can be moved

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

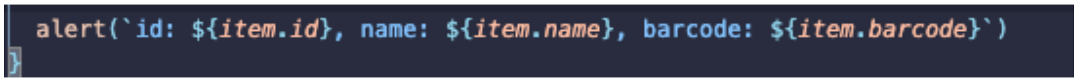
- Create a variable for the item as an object
- Create the id property from the query selector inner html for the data attribute for id
- Create the name property from the query selector inner html for the data attribute for name
- Create the barcode property from the query selector inner html for the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

• Create the barcode property from the query selector inner html for the data attribute for barcode



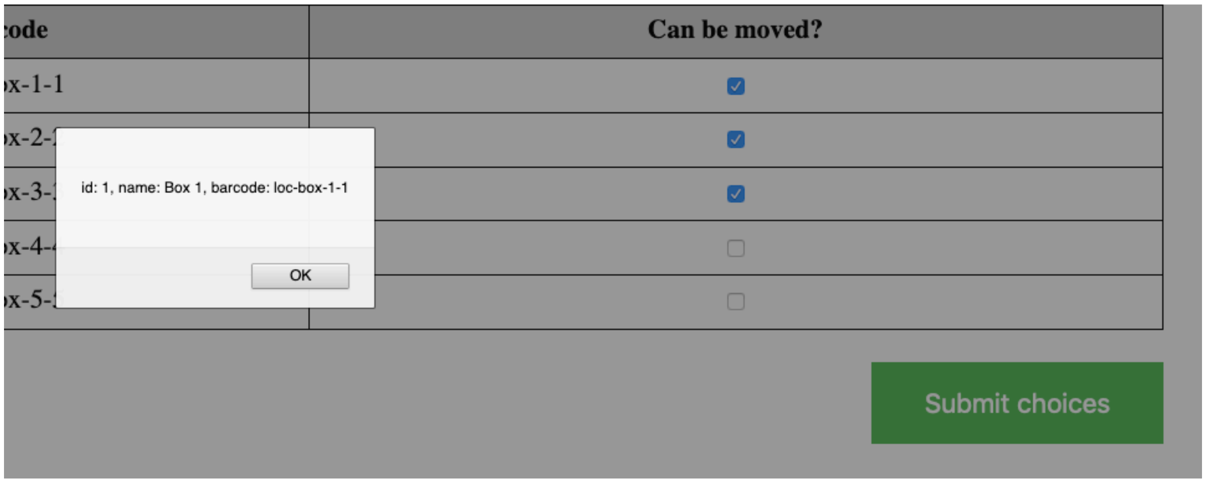
```
if (checkbox.checked) {

  const item = {}

  item.id = box.querySelector(`[data-attribute="id"]`).innerHTML
  item.name = box.querySelector(`[data-attribute="name"]`).innerHTML
  item.barcode = box.querySelector(`[data-attribute="barcode"]`).innerHTML

}
```

36. 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 …



```
  alert(`id: ${item.id}, name: ${item.name}, barcode: ${item.barcode}`)
}
```

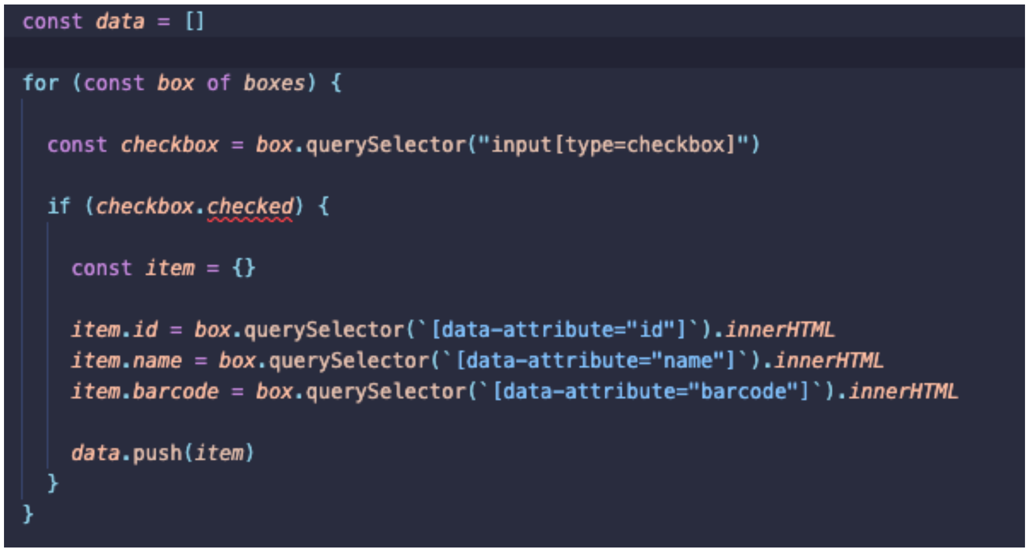
37. And then we can check this is working by (you guessed it) opening the browser!



ode	Can be moved?
x-1-1	☑
x-2-	☑
x-3-	☑
x-4-	☐
x-5-	☐

```
id: 1, name: Box 1, barcode: loc-box-1-1
```

38. 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.

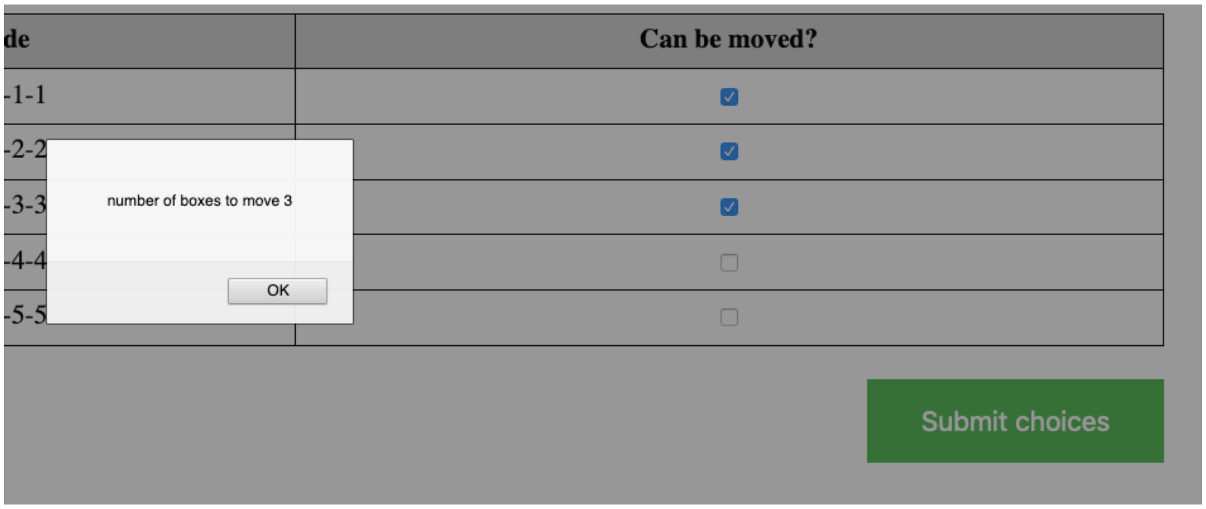


```
const data = []

for (const box of boxes) {

  const checkbox = box.querySelector("input[type=checkbox]")

  if (checkbox.checked) {

    const item = {}

    item.id = box.querySelector(`[data-attribute="id"]`).innerHTML
    item.name = box.querySelector(`[data-attribute="name"]`).innerHTML
    item.barcode = box.querySelector(`[data-attribute="barcode"]`).innerHTML

    data.push(item)
  }
}
```

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

```
alert(`number of boxes to move ${data.length}`)
}
```

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



de	Can be moved?
-1-1	✓
-2-2	✓
-3-3	✓
-4-4	☐
-5-5	☐

number of boxes to move 3

41. 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.

42. 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!

```
localStorage.setItem('boxes', JSON.stringify(data))
}
```

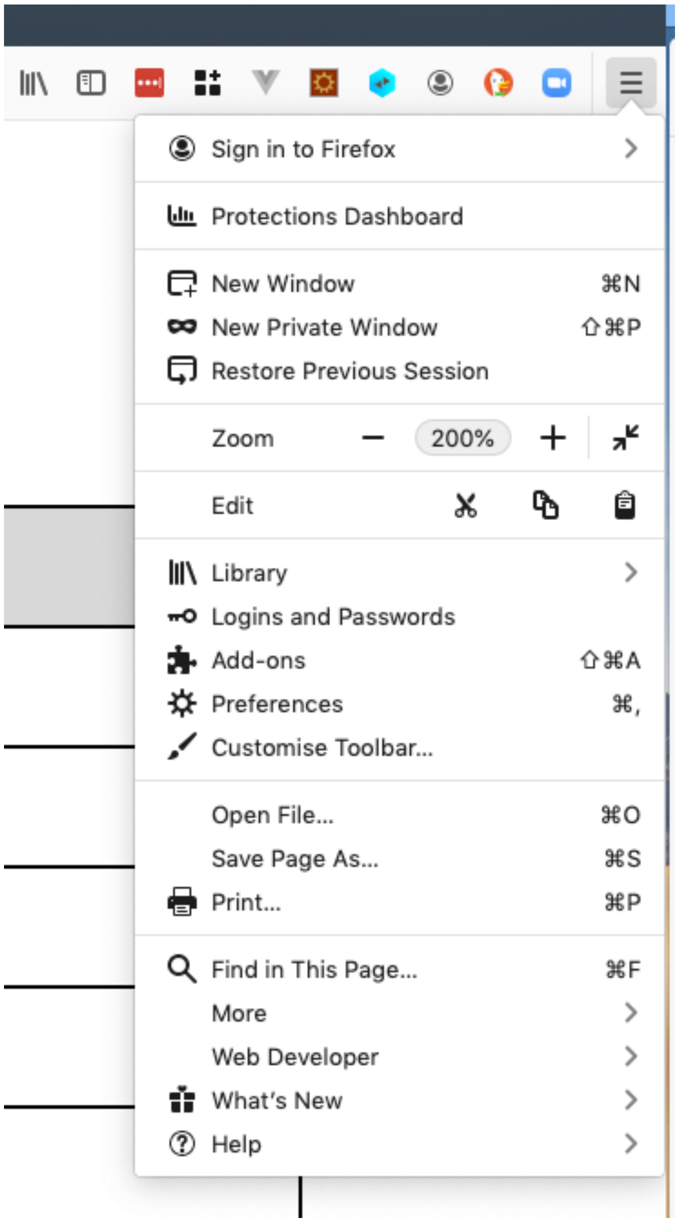
We will also add an alert to indicate to the user how many boxes have been moved into local storage just to indicate to the user that something has happened.

```
alert(`${data.length} boxes have been moved to local storage`)
```

43. 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.

44. 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.



45. 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.

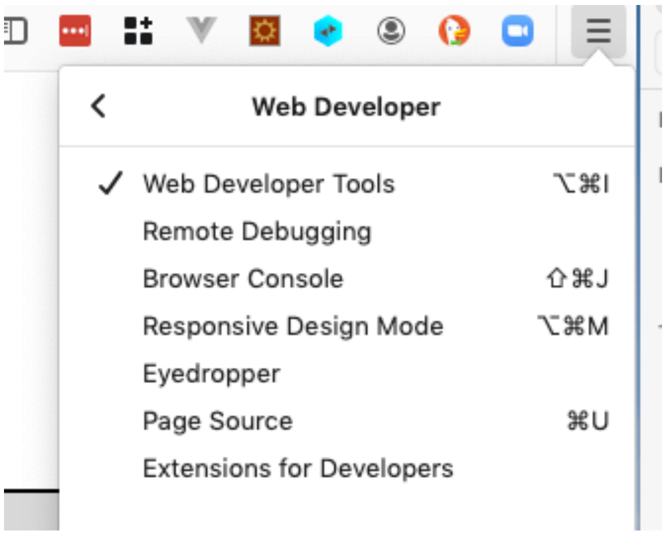
- **In IE11, you can see local storage in console on dev tools:**
- Show dev **tools** (press F12)
- Click "Console" or press Ctrl + 2.
- Type **localStorage** and press Enter.

- ## Open DevTools from Chrome's main menu

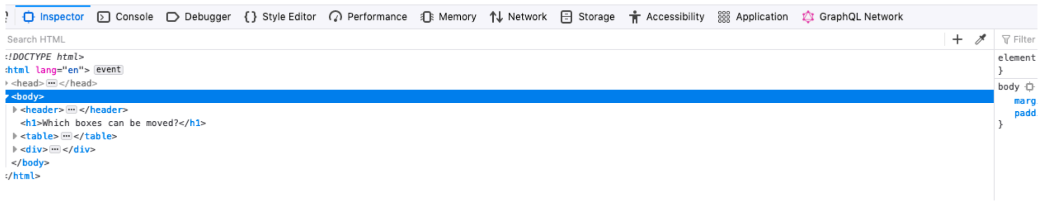
Open DevTools from Chrome's main menu

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

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


Web Developer
✓ Web Developer Tools ⌥⌘I
Remote Debugging
Browser Console ⇧⌘J
Responsive Design Mode ⌥⌘M
Eyedropper
Page Source ⌘U
Extensions for Developers

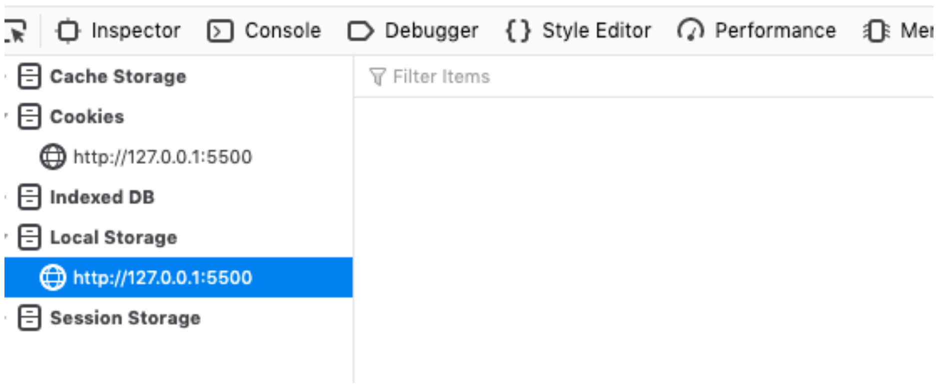
47. You should now have a new view in the bottom of your screen.



```
<!DOCTYPE html>
<html lang="en"> event
 <head>…</head>
 <body>
  <header>…</header>
  <h1>Which boxes can be moved?</h1>
  <table>…</table>
  <div>…</div>
 </body>
</html>
```

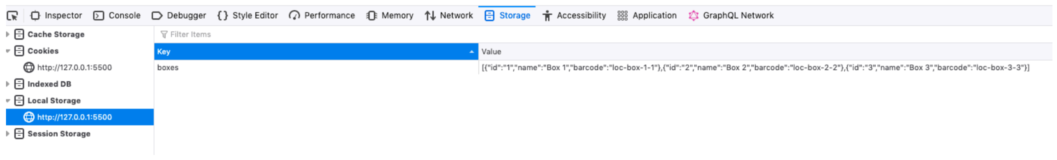
```
element
}
body
    marg
    padd
}
```

48. 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.



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

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



Key	Value
boxes	[{"id":"1","name":"Box 1","barcode":"loc-box-1-1"},{"id":"2","name":"Box 2","barcode":"loc-box-2-2"},{"id":"3","name":"Box 3","barcode":"loc-box-3-3"}]

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 …