Data Representation
Lab 2: functionality on the webpage

Lecturer: Andrew Beatty

Use the webpage created in Lab01 to create a webpage that has the functionality to:

1. Create a new Car **C**
2. View all the cars **R**
3. Update a Car **U**
4. Delete a car **D**

Show create

1. Hide the create form. Modify the surrounding the form so that it is hidden. Give this an id so that it can be discovered in javaScript.

```
<div id='createUpdateForm' style="display: none">
```

2. Give the create button and the table ids so that they can be discovered by javaScript.

```
<button id="showCreateButton">
…..
<table class="table" id="carTable">
```

3. Write a function called showCreate() that hides the create button and the table, and shows the createUpdateForm

```
function showCreate(){
        document.getElementById('showCreateButton').style.display="none"
        document.getElementById('carTable').style.display="none"
        document.getElementById('createUpdateForm').style.display="block"
    }
```

4. Call the function from when the createButton is clicked.

```
<button id="showCreateButton" onclick="showCreate()">
```

5. Add ids to the spans that surround the words create and update and the two buttons in the form

```
<span id="createLabel">Create a</span> <span id="updateLabel">update</span> Car
….
<span><button id="doCreateButton" onclick="doCreate()">Create</button></span>
<span><button id="doUpdateButton" onclick="doUpdate()">Update</button></span>
```

6. Add code to showCreate() that shows the create button and word and hides the update button and word. (test it)

```
document.getElementById('createLabel').style.display="inline"
document.getElementById('updateLabel').style.display="none"

document.getElementById('doCreateButton').style.display="block"
document.getElementById('doUpdateButton').style.display="none"
```

Show Update 1

7. Create a showUpdate(buttonElement) function, that shows the form. This function should show the update button and word and hide the create button and word.
8. Call this function from each of the update buttons in the table, "this" is the element that is activated

```
<td><button onclick="showUpdate(this)">Update</button></td>
```

9. We will be coming back to this function later

Do Create

10. Create a doCreate() function and call it from the create button. (test it with a console.log .

```
<td><button id="doCreateButton" onclick="doCreate()">Create</button>
```

```
function doCreate(){
}
```

11. Create a function called showViewAll that shows the create button and the table and hides the createUpdateForm.

12. Call this function from doCreate()

```
function doCreate(){
  showViewAll()
}
```

13. When you test your code you will notice that the data is still in the form when you click create a second time, we should clear this data. Create a function called clearForm. We will use the querySelector to find the inputs, instead of giving them all ids. (the disabled= false is for later).

```
function clearForm(){
     var form = document.getElementById('createUpdateForm')

     form.querySelector('input[name="reg"]').disabled = false
     form.querySelector('input[name="reg"]').value  =''
     form.querySelector('select[name="make"]').value=''
     form.querySelector('input[name="model"]').value=''
     form.querySelector('input[name="price"]').value=''
 }
```

14. Call this from the doCreate.

```
function doCreate(){
     // other code here
     clearForm()
     showViewAll()
 }
```

15. In do create read all the values from the form and store them in a car Object, we can test that this works by doing a console.log of the car Object.

```
     var form = document.getElementById('createUpdateForm')

     var car = {}
     car.reg = form.querySelector('input[name="reg"]').value
     car.make = form.querySelector('select[name="make"]').value
     car.model = form.querySelector('input[name="model"]').value
     car.price = form.querySelector('input[name="price"]').value
     console.log(JSON.stringify(car))
```

16. We will now need to add the car to the table, there is a lot to be done with this. Make a function called addCarToTable and call it from the doCreate.
17. In the function find the table and add a row to it at the end. And add a cell to that row and add the car reg to that cell (test this)

```
function addCarToTable(car){
        var tableElement = document.getElementById('carTable')
        var rowElement = tableElement.insertRow(-1)
// set attribure here
        var cell1 = rowElement.insertCell(0);
        cell1.innerHTML = car.reg

}
```

18. Add the rest of the data into the table

```
 function addCarToTable(car){
        var tableElement = document.getElementById('carTable')
        var rowElement = tableElement.insertRow(-1)
        // set attribure here
        var cell1 = rowElement.insertCell(0);
        cell1.innerHTML = car.reg
        var cell2 = rowElement.insertCell(1);
        cell2.innerHTML = car.make
        var cell3 = rowElement.insertCell(2);
        cell3.innerHTML = car.model
        var cell4 = rowElement.insertCell(3);
        cell4.innerHTML = car.price
        var cell5 = rowElement.insertCell(4);
        cell5.innerHTML = '<button onclick="showUpdate(this)">Update</button>'
        var cell6 = rowElement.insertCell(5);
        cell6.innerHTML = '<button onclick=doDelete(this)>delete</button>'

    }
```

19. test this the car should be added to the table when you click the do create button

show update2

20. When we are doing the update we would like the form populated with the details of the car we are updating, with that in mind we will need to do two things
    a. Read the car data from the current row,
    b. Populate the form with that car data.
21. Write a function called getCarFromRow, that takes in a Row element and returns a car object.

```
function getCarFromRow(rowElement){
       var car ={}
       car.reg  = rowElement.cells[0].firstChild.textContent
       car.make = rowElement.cells[1].firstChild.textContent
       car.model = rowElement.cells[2].firstChild.textContent
       car.price = rowElement.cells[3].firstChild.textContent
       return car
   }
```

22. Write a function called populateFormWithCar, which takes in a car

```
       function populateFormWithCar(car){
       var form = document.getElementById('createUpdateForm')
       form.querySelector('input[name="reg"]').disabled = true

       form.querySelector('input[name="reg"]').value  = car.reg
       form.querySelector('select[name="make"]').value= car.make
       form.querySelector('input[name="model"]').value= car.model
       form.querySelector('input[name="price"]').value= car.price
 }
```

23. Call these functions in showUpdate() so that the form is populated from the table. The rowElement is the grandparent of the buttonElement that was passed in as a parameter

```
function showUpdate(buttonElement){
….// other code here

       var rowElement = buttonElement.parentNode.parentNode
       // these is a way of finding the closest <tr> which would safer, closest()

       var car = getCarFromRow(rowElement)
       populateFormWithCar(car)
}
```

doUpdate

24. When the user clicks the update button on the form we will need to do two actions.
    a. Read the car from the form (like we did on doCreate)
    b. Modify the row that contains the car, to do this we will need to find the row, one solution is to add an id to each row, say the reg. You will also need to modify addToTable, later

```
                <tr id="191 Mo 123">
                        <td>191 Mo 123</td>
                        <td>Ford</td>
                        <td>Modeo</td>
                        <td>25,000</td>
                        <td><button onclick="showUpdate(this)">Update</button></td>
                        <td><button onclick="doDelete(this)">Delete</button></td>

                </tr>
                <tr id="12 G 123">….
```

25. Create a function called getCarFromForm, that returns a car

```
function getCarFromForm(){
        var form = document.getElementById('createUpdateForm')
        var car = {}
        car.reg = form.querySelector('input[name="reg"]').value
        car.make = form.querySelector('select[name="make"]').value
        car.model = form.querySelector('input[name="model"]').value
        car.price = form.querySelector('input[name="price"]').value
        console.log(JSON.stringify(car))
        return car
```

26. Create a function called setCarInRow that takes the car and rowElement and populated the row with the car (the opposite to getCarFromRow)

```
function setCarInRow(rowElement, car){
        rowElement.cells[0].firstChild.textContent= car.reg
        rowElement.cells[1].firstChild.textContent= car.make
        rowElement.cells[2].firstChild.textContent= car.model
        rowElement.cells[3].firstChild.textContent= car.price
    }
```

27. Put these together in doUpdate

```
function doUpdate(){
        var car = getCarFromForm();
        var rowElement = document.getElementById(car.reg)
        setCarInRow(rowElement,car)

        clearForm()
        showViewAll()
}
```

Delete

28. Put a doDelete(this) function into each of the delete buttons (code is above in 24b
29. We will use the deleteRow function which takes in the index of the row you wish to delete so we need to find the row and get its index. Write the code for doDelete.

```
function doDelete(r){
        var tableElement = document.getElementById('carTable')
        var index = r.parentNode.parentNode.rowIndex;
        tableElement.deleteRow(index);
 }
```

Fix bugs

30. Test the code and notice that new cars are not being updated, if you inspect the DOM you will notice that they do not have an id in the row like the other rows. In addCarToTable add a like that will set the id attribute of the new row to the reg

```
rowElement.setAttribute('id',car.reg)
```