GUI Lab 18 – JavaScript Arrays

Exercise 1:

Create an array containing 4 values as shown below:

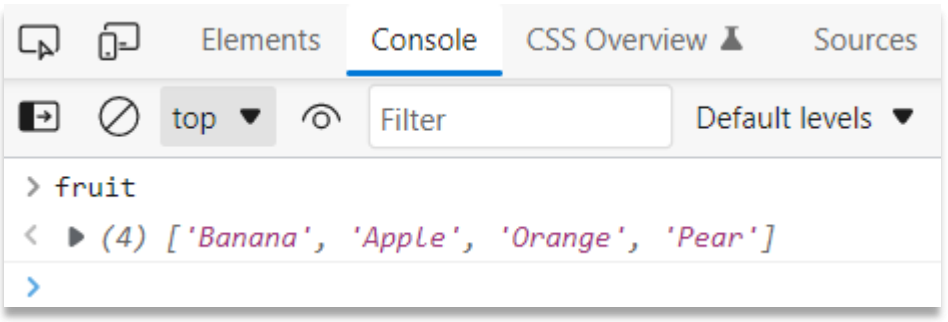
```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <title>Arrays</title>
</head>

<body>
    <script>
        let fruit = ["Banana", "Apple", "Orange", "Pear"];
    </script>
</body>
</html>
```

Output the array using an alert to confirm that it was created correctly

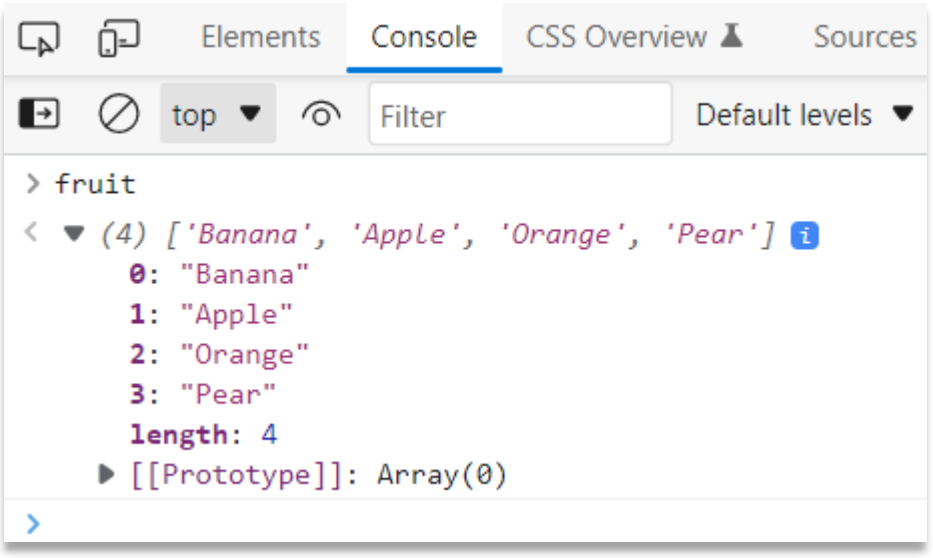
```
alert(fruit);
```

Open your page in a web browser – you should see the array output via an alert.

View the array using the developer console. Type in the name of the array at the console prompt:



Expand the array by clicking on the arrow beside the array name and examine the array structure:



Note that this can be useful to examine the status of an array to determine the index of each array value. For example, the value “Pear” is in index 3.

Output index 3 using an alert:

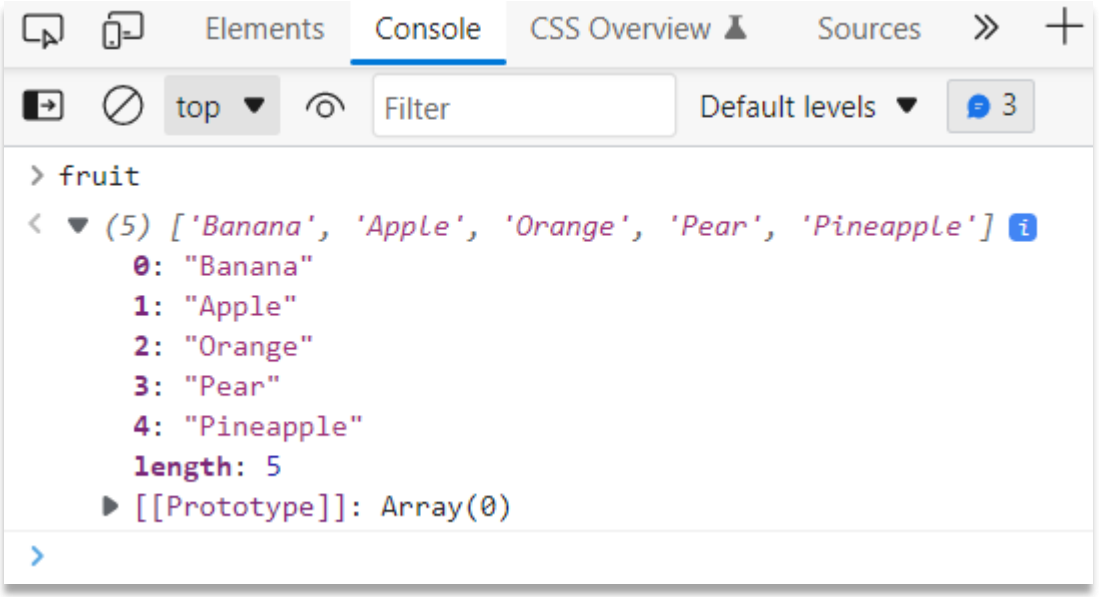
```
alert(fruit[3]);
```

This should only output the 4th value in the array at index 3, “Pear”.

Add a new value to the array with the following line:

```
fruit[4] = "Pineapple";
```

Confirm that this was added to the array by examining the array contents in the developer console:



The array should now contain a fifth element (in index 4), “Pineapple.”

Add new items to the array using the push() method:

```
fruit.push("Plum");
```

Output the array again to confirm the new value was added.

Use the length property to determine how many elements are in the array:

```
alert(fruit.length);
```

This should output the number of elements – 6.

Note that the first element begins with zero, therefore the last index is always the length minus 1.

Exercise 2:

Create a page as shown below:

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <title>Arrays</title>
</head>

<body>
    <script>
        let fruit = ["Banana", "Orange", "Apple", "Pear"];
    </script>
</body>
</html>
```

Amend your HTML to include the following on the page:

Add JavaScript code to your page so that when the user enters a fruit name into the input box and presses the button, the value is added to the array. You can use the push method to achieve this.

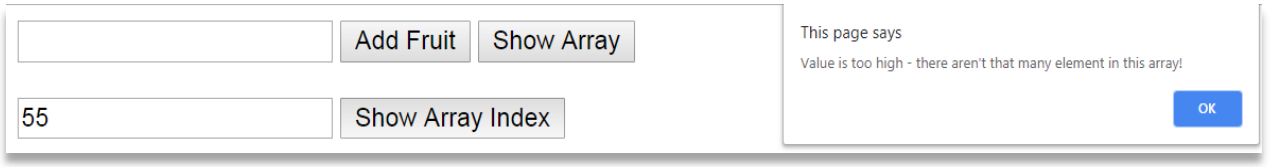
To check if your button is working, you can use an alert or the developer console to confirm if the array is being updated.

Add an additional button to your HTML as shown below:

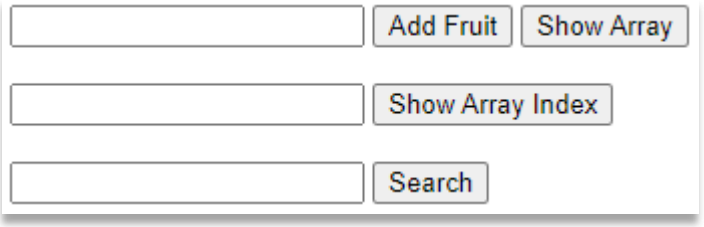
When this button is clicked, the array contents should be output to the browser window as shown below:

Amend the page to include a box which allows the user to specify an index number which will display the value stored at that index:

Add input validation to your page so that it will display a message if a value that is too high is entered in the search box:



Add a feature to search for a value entered into an input box:



We can use the indexOf method for this. Display an alert message indicating if the value is found or not.

Example:

fruit.indexOf("Plum");

will search the array for the value "Plum" and return its index. If -1 is returned then it cannot find the value.

Try this on your page using the developer console:

Check the current state of your array:

```
> fruit
< ▶ (4) ['Banana', 'Orange', 'Apple', 'Pear']
>
```

Then perform a search of the array using the following structure:

arrayname.indexOf("value to search for").

For example:

```
> fruit.indexOf("Apple")
< 2
>
```

This returns the index location of "Apple" (Note that this is case sensitive).

If you search for a non-exsitant value, the method will return a "-1":

```
> fruit.indexOf("abc")
< -1
>
```

Add a feature to delete an array element. Let's assume that we wish to remove the 3rd element (apple, which is in index 2) completely. We can use the splice() method to achieve this. Using the splice method we can specify on what index we want the remove to occur, and how many elements we wish to remove. For this exercise we will remove 1 element.

Therefore, to remove "apple" we will use the following:

```
fruit.splice(2,1);
```

After running this code, confirm what is contained in the array:

```
> fruit
< ▶ (3) ['Banana', 'Orange', 'Pear']
>
```

It should show that the array no longer contains the value "Apple".

Add a button to your page that will allow the user to specify an index and remove it from the array:

Exercise 3:

Create an array which contains some shopping list items and include it in the following web page:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Exercise 3</title>
  </head>
  <body>
    <h1>Shopping List</h1>
    <ul id="items"></ul>
    <script>
        let list = ["Bread", "Milk", "Tea", "Biscuits", "Crisps"];
    </script>
  </body>
</html>
```

Write JavaScript code to add the items from the list array to the unordered list on your webpage. There is more than one approach you could use to achieve this. One approach is you could use a loop to access each element of the array e.g.

```
for(let i = 0; i < list.length; i++){
```

Then build up a **string** with the list item elements (<li>) assigned the values of the array e.g.

```
output += "<li>"+list[i]+"</li>";
```

Then assign that **string** to the innerHTML property of the unordered list.

Example output as follows:

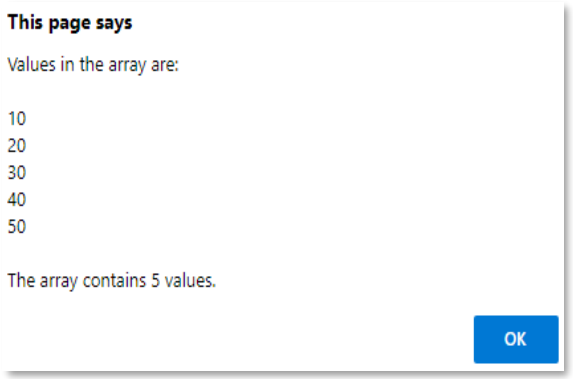
Exercise 4:

Consider the following array:

```
let numbers = [10, 20, 30, 40, 50];
```

Write JavaScript code to access each element of this array and output each element of the array to an alert box. Your output should also display the array length. Use the array length property to obtain this value.

An example output is shown on the next page:

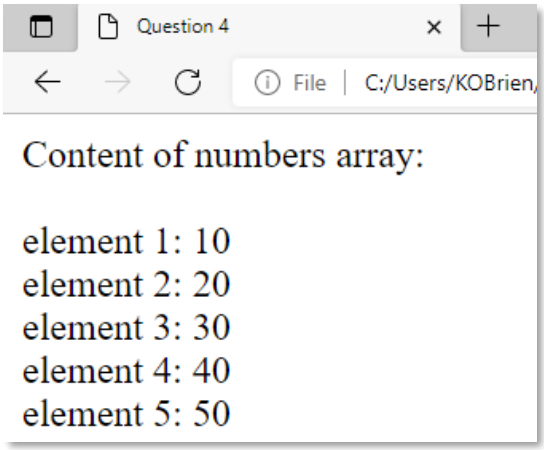


Exercise 5:

Consider the following array:

```
let numbers = [10, 20, 30, 40, 50];
```

Write JavaScript code which uses a loop to output each element of this array to the browser screen. Example output is as follows:

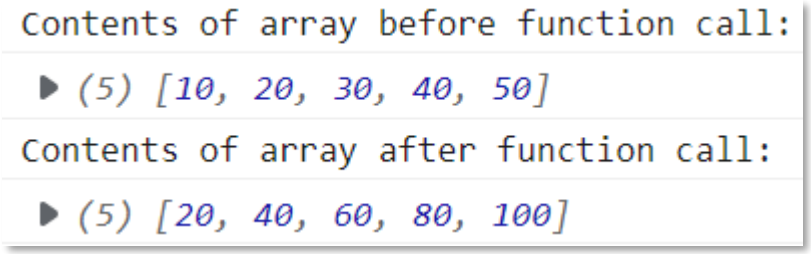


Exercise 6:

Write JavaScript code which invokes a function, doubleUp(array). The array parameter should be passed the following array:

```
let numbers = [10, 20, 30, 40, 50];
```

The function should multiply each of the values in the array by 2 and return the updated array. The contents of the updated array should be displayed in the console – use console.log and access the console in the browser by pressing the F12 key to see the output. Example output is as follows:



Exercise 7:

Write JavaScript code to output each of the elements of the following multidimensional array.

```
let numbers = [[1, 2, 3, 4], [5, 6, 7, 8]];
```

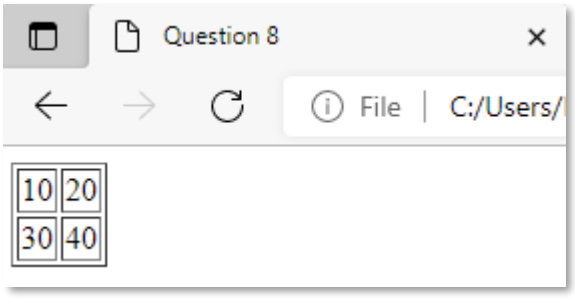
This array contains 2 rows with 4 columns in each row (a 2 by 4 2D array). The solution must access the contents of the arrays using a nested for loop. Output the contents of this array using an alert box. Example output is as follows:

Exercise 8:

Consider the following array:

```
let numbers = [[10, 20], [30, 40]];
```

Write JavaScript code to access each element of this array and output each element of the array to a HTML5 table in the browser:



To create a border on the table, use the following tag <table border='1'>. There are a number of ways of accessing each element of this two-dimensional array. One way would be to implement a nested loop to access each element and build up the table structure.

Exercise 9:

Create an array named deck that models a deck of cards. For example, "1_of_diamonds" represents the ace of diamonds, "2_of_diamonds" represents the 2 of diamonds, up to "13_of_diamonds", which represents the King of diamonds. The suits clubs, hearts and spades are represented in a similar manner. All these elements should be in a single array.

Confirm the array is created properly by outputting the entire contents of the array to the page, or by viewing it in the developer console.

Shuffle the deck

The deck can be shuffled using the function below.

```
function shuffle(){
    for (i=1; i<520; i++){
            randomNumber = Math.floor((Math.random() * 51));
            bottomCard = deck.pop();
            deck.splice(randomNumber, 0, bottomCard);
      }
}
```

Confirm the array has been shuffled by outputting the entire contents of the array to the page.

Using the pop method, get the last card in the pack and output it to the page.

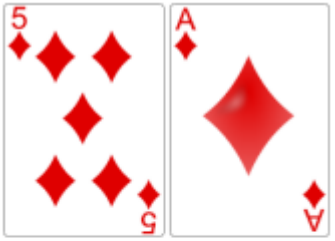
Example:

deck.pop()

Will return the last element of the array deck.

Now using the card images provided on Moodle (download card_images.zip and unzip this file), that outputs the last element in the deck as a card on the webpage. For example:

Finally, create a simple game of 21, where there are 4 cards taken from the deck and displayed on the screen – 2 cards for the dealer, and 2 cards for the player. The number of cards left in the deck after each deal should be displayed. You can use a button with a click event to deal the cards. Example output is as follows:



Cards left in deck: 36