arrays

const

index, array.length

methods (push(), sort(), pop(),)

nested / 2D / matrix arrays

arrays

Arrays are variables that can hold more than one value at a time. Arrays have a datatype of **object**. Here are some key things to know about arrays:

- array items exist as a list, surrounded by square brackets: `[item1, item2, item3]`
- each item is assigned a position number, or **index**, with the first item at index 0
- array items can be of any data type
- items of different data types can be in the same array: `[ 'hola', 38, true, ['apple, 'banana'] ]`

using const to declare an array

As detailed in a previous lesson, primitive variables (strings, numbers, booleans) declared with **const** cannot be modified (changed) in any way.

1. As a recap, declare a constant with `const`, and try to change it. Cannot be done:

```
const LITERS_PER_GALLON = 3.78;
LITERS_PER_GALLON = 4; // ERROR: Assigment to constant variable
```

const for objects

However, arrays declared with `const` *can* be modified. Items can be changed, added or deleted. But the array object itself cannot be *mutated* to a different datatype. So, with *const*, once an array, always an array.

how to declare an array

To declare an array, start with the `let` or `const` keyword, followed by a name. We will start with **let** in order to later prove our point about mutability. The array is wrapped in square brackets, with items separated by a comma.

It is a good practice to pluralize array names so: *fruits* not *fruit*. You can also add *Arr* to make it crystal clear that this is an array: *fruitsArr*.

2. Declare an array called `fruitsArr`, with three items:

```
let fruitsArr = ['apple', 'banana', 'cherry'];
console.log(fruitsArr);
```

3. Check the Console. Expand the arrow to see the numbered items, with 'apple' at index 0.

index

Array items are stored in a numeric order, called **index**.

- The first item in an array is at index 0, so if there are 3 items in an array, the last item is at index 2.
- Use square bracket syntax to access items: `array[0]`

4. Get the first item at index 0 and the last item at index 2:

```
console.log(fruitsArr[0]); // apple
console.log(fruitsArr[2]); // cherry
```

setting array values by index

To set the value of an array item, refer to it by index:

5. Replace the first item, 'apple', with 'peach':

```
fruitsArr[0] = 'peach';
console.log(fruitsArr);
// ['peach', 'banana', 'cherry']
```

One way to add an item at the end of the array is to assign a value after the last item:

6. Add 'mango' to the end of the array, at index 3:

```
fruitsArr[3] = 'mango';
console.log(fruitsArr);
// ['peach', 'banana', 'cherry', 'mango']
```

mutating an array (changing its datatype

One downside of `let` for arrays is that it does not protect the data type. You could inadvertently change the array to a string or any other datatype. Use `const` to prevent such an accident from occurring.

7. Change *fruitsArr* to a string. Just like that, no more array:

```
fruitsArr = "fresh";
console.log(fruitsArr); // fresh
```

8. Declare another fruits array, this time with `const`. We need a new name, as `const` cannot redeclare an existing variable:

```
const fruits = ['apricot', 'pear', 'kiwi', 'grape'];
```

9. Try to mutate this `const` array into a string. You get an error:

```
const fruits = "ripe";
// ERROR: Assignment to constant variable
```

array.length property

The `length` property of an array returns the number of items in the array.

10. Get the length of the array:

```
console.log(fruits.length); // 4
```

array[arrray.length-1]

- The first item in an array is at: **array[0]**.
- The last item is at: **array[array.length-1]**.

11. Use `length-1` to access the last item in the array:

```
let lastFruit = fruits[fruits.length - 1];
console.log(lastFruit); // grape
```

random array items

To get a random item from an array, generate a random integer within the range of the array length, and pass that number to the array square brackets:

12. Get a random fruit from the *fruits* array.

```
let randInt = Math.floor(Math.random() * fruits.length);
let randFru = fruits[randInt];
```

```
console.log(randFru);
```

13. Run it a few times to see that the fruit keeps changing.

array methods

The array object has numerous methods, some of which we will learn about now:

array.push(item)

We used `fruits[3]` to add to the end of the array. But what if the length of the array is unknown? Better is to use the `push()` method, which adds an item to the end without needing to know how many items are in the array.

14. Using the **push()** method, add 'lime' to the end of the array.

```
fruits.push('lime');
console.log(fruits);
// ['apricot', 'pear', 'kiwi', 'grape', 'lime']
```

declaring a new empty array

15. Use the **sort()** method to put the string items of an array in alphabetical order:

```
fruits.sort();
console.log(fruits);
// ['apricot', 'grape', 'kiwi', 'lime', 'pear']
```

array.pop()

The **pop()** method removes the last item from the array and returns it, so the item can be saved by setting `pop()` equal to a variable.

16. Using the **pop()** method, remove the last item and save it to a variable, `popped`:

```
let popped = fruits.pop();
console.log(popped); // pear
console.log(fruits);
// ['apricot', 'grape', 'kiwi', 'lime']
```

An array can be declared with just a pair of empty curly braces--no items.

17. Declare a new, empty array as a pair of square brackets:

```
const veggies = [];
```

18. Use the push() method to populate the array:

```
veggies.push('carrot');
veggies.push('celery');
console.log(veggies);
// ['carrot', 'celery']
```

2D maxtrix arrays

An array can have arrays for its items. The terms to describe such an array include: **matrix, 2D array** and **nested array**

19. Make a 3x3 matrix of three items, each an array of three items.

```
const nestedArr = [[1,2,3], [4,5,6], [7,8,9]];
console.log(nestedArr.length);
console.log(nestedArr);
console.log(nestedArr[0]); // [1,2,3]
```

20. Use double square brackets: the first to get the inner array, the second to get the inner array item:

```
console.log(nestedArr[0][0]); // 1
console.log(nestedArr[1][1]); // 5
console.log(nestedArr[2][2]); // 9
```

21. Represent a tic-tac-toe board, where all 9 squares start out with a value of null:

```
let ticTacToe = [
    [null, null, null],
    [null, null, null],
    [null, null, null]
];
```

22. Then the game starts; "X" chooses the middle square, and "O" chooses the lower left square:

```
ticTacToe[1][1] = "X";
ticTacToe[2][0] = "O";
```

23. Log the result: it's a tic-tac-toe game in progress:

```
console.log(ticTacToe);
/*
(3) [Array(3), Array(3), Array(3)]
0: (3) [null, null, null]
1: (3) [null, 'X', null]
2: (3) ['O', null, null]
*/
```

END: Lesson 04.01

NEXT: 04.01 Lab Lesson 04.02