Array Methods

push(), pop(), sort()

unshift(), shift(), concat()

splice(), slice(), includes()

indexOf(), lastIndexOf()

join(), flat(), reverse()

Array methods are called on arrays and perform operations. We looked at a few array methods in a previous lesson, namely: `push()`, `pop()` and `sort()`. Let's recap those three before moving on:

1. Declare an array of a few items, and use `push()` to add a couple of items at the end:

```
const fruits = ['kiwi', 'cherry', 'banana'];
fruits.push('orange');
fruits.push('grape');

console.log(fruits);
// ['kiwi', 'cherry', 'banana', 'orange', 'grape']
```

2. Remove the last item using `pop()`. The method returns the popped item, so save that to a variable:

```
let poppedItem = fruits.pop();
console.log(poppedItem); // grape

console.log(fruits);
// ['kiwi', 'cherry', 'banana', 'orange']
```

3. Arrange the items in alphabetical order with `sort()`:

```
fruits.sort();

console.log(fruits); // ['banana', 'cherry', 'kiwi', 'orange']
```

Now, for some more array methods:

- **unshift()** -- add item to beginning
- **shift()** -- remove item from beginning
- **concat()** -- combine two or more arrays
- **splice()** -- remove or swap out items
- **slice()** -- make a new array from a range of items
- **includes()** -- looks for item and returns true or false
- **indexOf()** -- returns index of first matching item
- **lastIndexOf()** -- returns index of last matching item
- **join()** -- turn an array into a string
- **flat()** -- turn a matrix into a 1D array
- **reverse()** -- reverse the order of items

shift() and unshift()

- **unshift()** adds an item to the beginning of an array.
- **shift()** removes the first item and returns it.

To help remember which is which, "unshift" is a longer word than "shift", just as "push" is longer than "pop". The longer words make the array longer, while the shorter words make the array shorter.

4. Use **unshift()** to add an item to the beginning of the fruits array:

```
fruits.unshift('apple');

console.log(fruits);
// ['apple', 'banana', 'cherry', 'kiwi', 'orange']
```

5. Use **shift()** to remove and return the first item, saving that to a variable:

```
let shiftedItem = fruits.shift();

console.log(shiftedItem); // apple
console.log(fruits);
// ['banana', 'cherry', 'kiwi', 'orange']
```

concat()

As the name implies, **concat()** concatentates (combines) two or more things, in this case arrays. You call **concat()** on one array, and pass the method the other array(s) as its argument(s).

6. Declare three arrays, and then concat them into one:

```
const tropicalFruits = ['mango', 'kiwi', 'banana', 'pineappple'];

const citrusFruits = ['orange', 'lemon', 'lime', 'tangerine'];
```

```
    const blossomFruits = ['apple', 'peach', 'cherry', 'plum'];

    const fruitCocktailArr = tropicalFruits.concat(citrusFruits,
blossomFruits);

    console.log(fruitCocktailArr);
    // ['mango', 'kiwi', 'banana', 'pineappple', 'orange', 'lemon',
'lime', 'tangerine', 'apple', 'peach', 'cherry', 'plum']
```

7. Sort the fruit cocktail array. The `sort()` method changes the original array; it does not return a new array.

```
    fruitCocktailArr.sort();

    console.log(fruitCocktailArr);
    // ['apple', 'banana', 'cherry', 'kiwi', 'lemon', 'lime', 'mango',
'orange', 'peach', 'pineappple', 'plum', 'tangerine']
```

splice()

The **splice()** method removes or swaps out items at a specified index or range of indices. With **splice()**, you can remove or replace any item, or sequence of items, anywhere in the array.

- `splice()` takes two arguments: the index of the first item to remove, and the number of items to remove.
- `splice()` returns the removed item(s). If more than one item was removed, it returns an array.

8. Remove the item at index 9, which is 'pineapple':

```
    fruitCocktailArr.splice(9,1);

    console.log(fruitCocktailArr);
    // ['apple', 'banana', 'cherry', 'kiwi', 'lemon', 'lime', 'mango',
'orange', 'peach', 'plum', 'tangerine']
```

9. Starting at index 2, splice out four consecutive items, saving the result to a new array:

```
    let splicedItems = fruitCocktailArr.splice(2,4);

    console.log(splicedItems);
    // ['cherry', 'kiwi', 'lemon', 'lime']
    console.log(fruitCocktailArr);
    // ['apple', 'banana', 'mango', 'orange', 'peach', 'plum',
'tangerine']
```

To swap an item with `splice()`, pass in the new item as additional argument(s).

10. Using `splice()`, swap 'peach' for 'papaya':

```
    fruitCocktailArr.splice(4, 1, 'papaya');

    console.log(fruitCocktailArr);
    // ['apple', 'banana', 'mango', 'orange', 'papaya', 'plum',
'tangerine']
```

To swap more than one item with `splice()`, just add more arguments.

11. Replace 'banana' and 'mango' with 'blueberry' and 'strawberry'.

```
    fruitCocktailArr.splice(1, 2, 'blueberry', 'strawberry');

    console.log(fruitCocktailArr);
    // ['apple', 'blueberry', 'strawberry', 'orange', 'papaya', 'plum',
'tangerine']
```

splice for avoiding repeats of random array items

If you keep choosing items at random from an array, you will eventually get repeats. To avoid repeats, splice out each item as you go.

12. Pick two random fruits from `fruitCocktailArr`, and log them both:

```
    let r1 = Math.floor(Math.random() * fruitCocktailArr.length);
    let rFruit1 = fruitCocktailArr[r1];

    let r2 = Math.floor(Math.random() * fruitCocktailArr.length);
    let rFruit2 = fruitCocktailArr[r2];

    console.log(rFruit1, rFruit2);
```

13. Rerun the console.log until you get a repeat fruit. To rerun the console command, hit the Up Arrow and then hit Enter.

To avoid repeats, each time a random fruit is chosen, splice it out of the array with `splice(r, 1)`. The index, `r`, is the random item and the number of items to remove is 1.

14. Choose two random fruits again, this time splicing out the first fruit before picking the second one:

```
    r1 = Math.floor(Math.random() * fruitCocktailArr.length);
    rFruit1 = fruitCocktailArr[r1];
```

```
fruitCocktailArr.splice(r1, 1);

r2 = Math.floor(Math.random() * fruitCocktailArr.length);
rFruit2 = fruitCocktailArr[r2];
```

15. Rerun the console.log command repeatedly to verify that no repeats occur. Soon, the array will be empty and the results `undefined`:

```
console.log(rFruit1, rFruit2);
```

slice()

The **slice()** method is called on an array and takes two arguments: a starting and ending index.

- `slice()` returns a new array without affecting the original array.
- `slice()` end index is exclusive, so *not* included in the new array.

16. Starting at index 2 and ending at index 5 (exclusive), get a new an array of 3 items:

```
let fruitSlices = fruitCocktailArr.slice(2, 5);

console.log(fruitSlices);
// ['strawberry', 'orange', 'papaya']
```

If you omit the second argument, it slices from the start index (first argument) all the way to the end:

17. Starting at index 4, slice all the way to the end:

```
let slicedFruit = fruitCocktailArr.slice(4);

console.log(slicedFruit);
// ['papaya', 'plum', 'tangerine']
```

includes()

The **includes()** method is called on an array and returns true if its argument is found in the array, and false if it is not:

18. Call the `includes()` method to get one true and one false result:

```
console.log(slicedFruit.includes('plum'));
// true
```

```
    console.log(slicedFruit.includes('pear'));
    // false
```

indexOf()

The **indexOf()** method is called on an array and returns the index of the first instance of the argument. If it is not found, it returns -1.

19. Declare an array of reptiles and run the indexOf() method a few times:

```
    const reptiles = ['iguana', 'snake', 'turtle', 'snake', 'gekko',
'snake', 'lizard'];

    console.log(reptiles.indexOf('snake')); // 1
    console.log(reptiles.indexOf('turtle')); // 2
    console.log(reptiles.indexOf('Komodo dragon')); // -1
```

To specify a starting index, pass in a second argument. This skips earlier instances of the word:

20. Starting at index 2, get the position of the first snake after that:

```
    console.log(reptiles.indexOf('snake', 2)); // 3
```

lastIndexOf()

The **lastIndexOf()** method returns the index of the last occurrence of the argument. If it is not found, it returns -1:

```
    console.log(reptiles.lastIndexOf('snake')); // 4
    console.log(reptiles.lastIndexOf('Gila monster')); // -1
```

join()

The **join()** method is called on an array and returns a string of all the items, separated by commas. It does not change the array.

21. Join the reptiles into a long slithering string:

```
    let reptilesStr = reptiles.join();
    console.log('reptilesStr', reptilesStr);
    // iguana,snake,turtle,snake,gekko,snake,lizard
```

The join method can take a *delimiter* argument--character(s) that will appear between the items in the resulting string.

22. Put an asterisk surrounded by spaces between each fruit in the string:

```
    let starryFruits = fruitCocktailArr.join(' * ');
    console.log(starryFruits);
    // apple * blueberry * strawberry * orange * papaya * pineappple *
plum * tangerine
```

flat()

The **flat()** method takes a nested array (2D, matrix) as its argument and returns a flat, one-dimensional array. It does not change the original array.

23. Declare a 3x3 / 2D / nested array:

```
    const ticTacToe = [
        ['X', 'O', null],
        [null, 'X', 'O'],
        ['O', null, 'X']
    ];
```

24. Flatten the array:

```
    const flatArray = ticTacToe.flat();

    console.log(flatArray);
    // console.log('flatArr', flatArr);
    // ['X', 'O', null, null, 'X', 'O', 'O', null, 'X']
```

copying an array with **slice(0)

An array can be copied in a number of ways. One way is `slice(0)`. Starting at index 0 and going all the way to the end by omitting the second argument, it returns a new array which includes all items in the original.

25. Declare an array and make a copy of it:

```
    const animals = ['giraffe', 'leopard', 'ostrich', 'zebra',  'panda',
'moose', 'bison', 'aardvark', 'baboon', 'rhinoceros'];

    const animalsCopy = animals.slice(0);

    console.log(animalsCopy);
    // ['giraffe', 'leopard', 'ostrich' etc.]
```

reverse()

The **reverse()** method reverses the order of the items in an array. It is often combined with **sort()** to flip the sorted order:

26. Sort and then reverse the animals array:

```
    console.log(animals.sort());
    console.log(animals.reverse());
    // ['zebra', 'rhinoceros', 'panda', 'ostrich', 'moose', 'leopard',
'giraffe', 'bison', 'baboon', 'aardvark']
```

methods chaining

Methods can be called one after another, on the same line. This is known as *methods chaining*.

27. In one line of code, sort and then reverse the `animalsCopy` array:

```
    animalsCopy.sort().reverse();

    console.log('animalsCopy', animalsCopy);
```

END Lesson 04.03

NEXT: Lab 04.03 Lesson 04.04