Session 18

Mohamed Emary

May 27, 2024

1 Built-in Array Methods

1.1 `push`

Suppose you have this array:

```
var colors = ["red", "orange", "yellow"];
```

And you want to add another color to the end of the array. You can use this way:

```
colors[3] = "green";
console.log(colors); // ["red", "orange", "yellow", "green"]
```

However, there is a better way to do this. You can use the `push` method:

```
var colors = ["red", "orange", "yellow"];
colors.push("green");
console.log(colors); // ["red", "orange", "yellow", "green"]
```

The `push` method is an array method that adds an element to the end of an array, and returns the new length of the array.

You can also use `push` to add multiple elements to an array:

```
var colors = ["red", "orange"];
var newLen = colors.push("blue", "indigo", "violet");
console.log(colors); // ["red", "orange", "blue", "indigo", "violet"]
console.log(newLen);    // 5
```

1.2 `unshift`

`unshift` is similar to `push`, but it adds an element to the beginning of an array instead of the end, and returns the new length of the array.

```
var colors = ["red", "orange", "yellow"];
var newLen = colors.unshift("green");
```

```
console.log(colors); // ["green", "red", "orange", "yellow"]
console.log(newLen);    // 4
```

You can also use `unshift` to add multiple elements to an array:

```
var colors = ["red", "orange"];
colors.unshift("blue", "indigo", "violet");
console.log(colors); // ["blue", "indigo", "violet", "red", "orange"]
```

1.3 `pop`

`pop` removes the last element from an array and returns that element.

```
var colors = ["red", "orange", "yellow"];
var removed = colors.pop();
console.log(removed); // "yellow"
console.log(colors);  // ["red", "orange"]
```

1.4 `shift`

`shift` removes the first element from an array and returns that element.

```
var colors = ["red", "orange", "yellow"];
var removed = colors.shift();
console.log(removed); // "red"
console.log(colors);  // ["orange", "yellow"]
```

Hovering on functions

When you hover with the mouse cursor over a function in VS Code, you can see a brief description of what the function does. This can be helpful if you're not sure what a function does or how to use it.

To know the length of an array, you can use the `length` ***property***:

It's a property, because arrays are objects, and `length` is a property of the array object.

```
var colors = ["red", "orange", "yellow"];
console.log(colors.length); // 3
```

`length` is always higher than the highest index in the array by 1 since it starts from 1 and index starts from 0, so if you have an array with 3 elements, the highest index is 2, and the length is 3.

1.5 `splice`

`splice` can add or remove elements from an array, and returns the removed elements.

It takes three arguments:

1. The index at which to start changing the array.
2. The number of elements to remove.
3. The element***s*** to add.

So this is its syntax:

```
array.splice(start, deleteCount, item1, item2, ...)
```

Example on deleting elements with `splice`:

```
var colors = ["red", "orange", "yellow", "green", "blue"];
// To delete yellow and green
var removed = colors.splice(2, 2); // From index 2, delete 2 elements
console.log(colors);  // ["red", "orange", "blue"]
console.log(removed); // ["yellow", "green"]

colors = ["red", "orange", "yellow", "green", "blue"];
// To delete all elements starting from index 2
removed = colors.splice(2);  // From index 2, delete all elements
console.log(colors);  // ["red", "orange"]
console.log(removed); // ["yellow", "green", "blue"]

colors = ["red", "orange", "yellow", "green", "blue"];
// This have no effect on the array
removed = colors.splice(2, 0);  // From index 2, delete 0 elements
console.log(colors);  // ["red", "orange", "yellow", "green", "blue"]
```

Example on adding elements with `splice`:

```
var colors = ["red", "orange"];
// To add yellow and black
colors.splice(1, 0, "yellow", "black"); // From index 1, delete 0 elements,
 ↪  add "yellow" and "black"
console.log(colors); // ["red", "yellow", "black", "orange"]


colors = ["red", "orange", "yellow"];
// To add green and blue and remove orange
colors.splice(2, 1, "green", "blue"); // From index 2, delete 1 element,
 ↪  add "green" and "blue"
console.log(colors); // ["red", "orange", "green", "blue"]
```

1.6 `slice`

`slice` takes two arguments: the start index and the end index (*not included*), and returns a new array with the elements between the start and end indexes.

Not included means that the last index is not included in the new array.

With `slice`, the original array is not affected.

```
var colors = ["red", "orange", "yellow", "green", "blue"];
var subColors = colors.slice(1, 3); // From index 1 to 3 (not included)
console.log(subColors); // ["orange", "yellow"]
```

1.7 `includes`

`includes` checks if an array includes a certain element, and returns `true` or `false`.

```
var colors = ["red", "orange", "yellow"];
console.log(colors.includes("yellow")); // true
console.log(colors.includes("purple")); // false
```

You can also specify a starting index from which to start searching:

```
var colors = ["red", "orange", "yellow", "green", "blue"];
console.log(colors.includes("orange", 2)); // false
```

Here the color `"orange"` already exists in the array, but since we specified the starting index as `2`, it starts searching from index `2`, and `"orange"` is at index `1`, so it returns `false`.

1.8 `indexOf` & `lastIndexOf`

`indexOf` returns the first index at which a given element can be found in the array, or `-1` if it is not present.

```
var colors = ["red", "orange", "yellow", "green", "blue"];
console.log(colors.indexOf("green")); // 3
console.log(colors.indexOf("purple")); // -1
```

If the array contains two similar elements, `indexOf` returns the index of the first one:

```
var colors = ["orange", "red", "yellow", "red"];
console.log(colors.indexOf("red")); // 1
```

What if you want to get the index of the last occurrence of an element? You can use `lastIndexOf`:

```
var colors = ["orange", "red", "yellow", "red"];
console.log(colors.indexOf("red")); // 1
console.log(colors.lastIndexOf("red")); // 3
```

Other Methods

There are many other methods that you can use with arrays. You can find the full list of array methods in the MDN Web Docs.

Some methods like `map`, `filter`, `reduce`, and `forEach` were introduced in ES6, and they are very useful when working with arrays. We will cover them in the next sessions.

2 Exercise

Try to find how many times a certain element appears in an array.

```
function getOccurrences(array, searchElement) {
  var indices = [];
  if (array.includes(searchElement)) {
    for (var i = 0; i < array.length; i++) {
      if (array[i] === searchElement) {
        indices.push(i);
      }
    }
    return indices;
  } else {
```

```
    return 0;
  }
}

var numbers = [1, 2, 3, 4, 1, 1, 1, 1];
console.log(getOccurrences(numbers, 1)); // [0, 4, 5, 6, 7]
```

3 CRUD Operations

CRUD stands for ***C***reate, ***R***ead, ***U***pdate, and ***D***elete. These are the four basic operations that can be performed on data. You can also add ***S*** for ***S***earch.

Any software application that works with data usually performs these operations.

When working making your website always finish the design first (the HTML and CSS), then add the functionality (JavaScript).

Also when solving a problem using JS divide it into smaller tasks, and solve each task separately.

Important:

Watch the CRUD example system videos from video 4 to 9.

When getting an element from the DOM using `document.getElementById("id")`, you can use `console.log(element)` it to make sure you got the right element.

When JS deals with HTML it converts the HTML tags to objects each with its own properties and methods and each attribute in the HTML tag is a property in the object, and you can manipulate these objects using JS.

```
var addBtn = document.getElementById("addBtn");

addBtn.onclick = addProduct; // IMPORTANT: Don't add () after the function
 ↪  name

function addProduct() {
  var prodName = document.getElementById("prodName").value;
  console.log(prodName);
};
```

In this code we assigned the `addProduct` function to the `onclick` event of the `addBtn` button, and we didn't add `()` after the function name, because we don't want to call the function immediately, we want to call it when the button is clicked.

The `addProduct` function gets the value of the input with the id `prodName` and logs it to the console.

Important Note:

You shouldn't put this line before the function:
`var prodName = document.getElementById("prodName").value;`
Because the input field will not have a value when the page loads, and you want to get the value when the button is clicked, as the user will press it after typing the product name.

4 Summary

In this session we learned about the following array methods:

- `push` adds an element to the end of an array, and returns the new length of the array.
- `unshift` adds an element to the beginning of an array, and returns the new length of the array.
- `pop` removes the last element from an array, and returns that element.
- `shift` removes the first element from an array, and returns that element.
- `splice` can add or remove elements from an array, and returns the removed elements.
- `slice` returns a new array with the elements between the start and end indexes.
- `includes` checks if an array includes a certain element, and returns `true` or `false`.
- `indexOf` returns the first index at which a given element can be found in the array, or `-1` if it is not present.
- `lastIndexOf` returns the index of the last occurrence of an element in an array.

We also learned about CRUD operations, and how to manipulate the DOM using JavaScript.