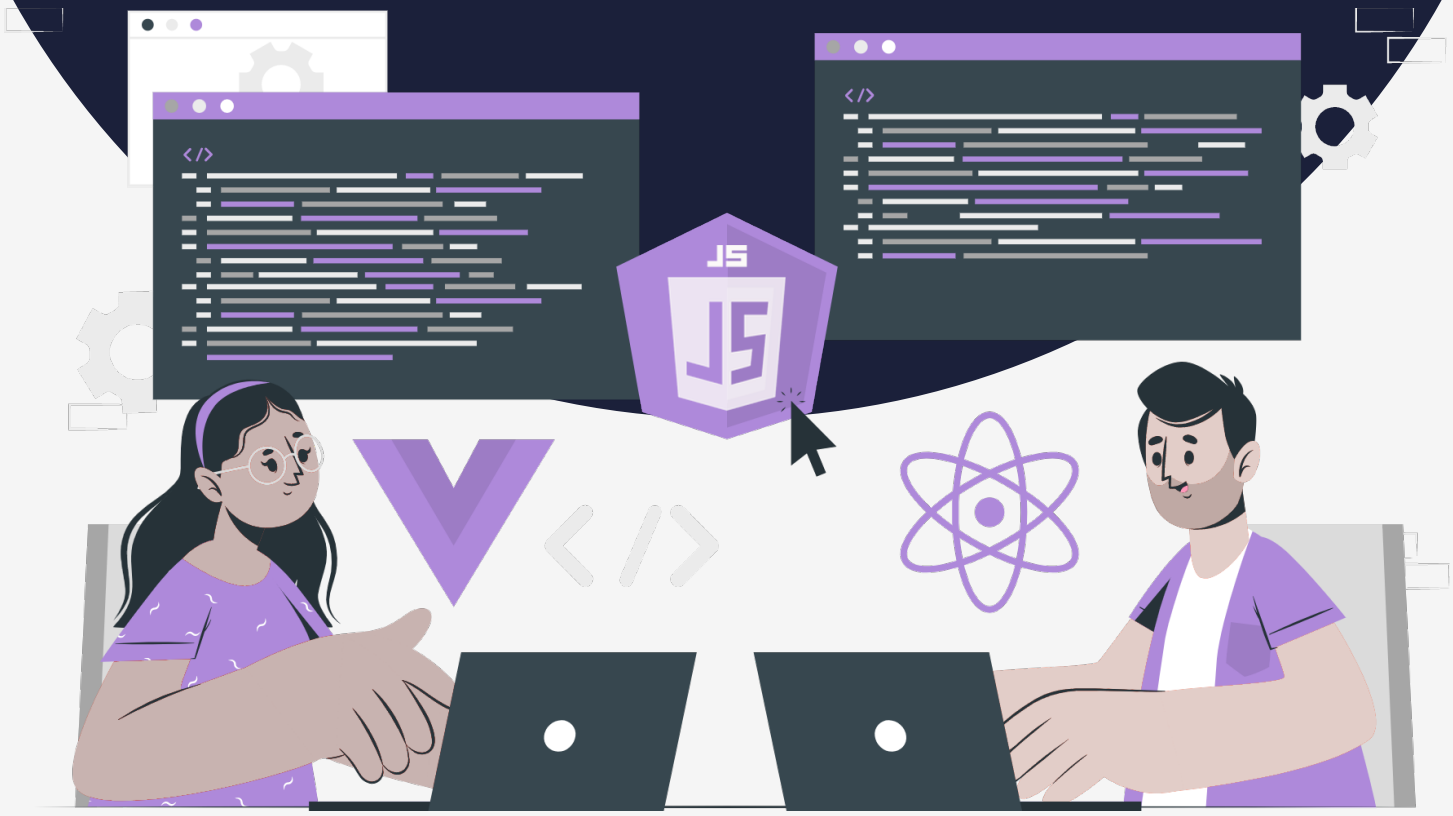
# Lesson:

# Array Methods




PW SKILLS

# Topics Covered:

- pop()
- push()
- shift()
- unshift()
- Concat()
- join()
- slice()
- splice()
- reverse()
- indexOf()

Array methods are built-in functions in JavaScript that allow you to perform various operations on arrays. They can be used to manipulate arrays in various ways, including adding and removing elements, sorting, transforming, and filtering the elements of an array.

**pop():**

The pop() method in JavaScript is used to remove the last element of an array and return the removed element. The method modifies the original array, reducing its length by one.

```
let fruits = ['apple', 'banana', 'cherry'];
let last = fruits.pop();
console.log(fruits); // Output: [ 'apple', 'banana' ]
console.log(last); // Output: 'cherry'
```

In the example above, fruits.pop() removed the last element 'cherry' from the fruits array and returned it. The fruits array now only contains two elements, 'apple' and 'banana'.

It's important to note that the pop() method modifies the original array, so if you need to keep the original array intact, you should make a copy of it before using pop().

**push():**

The push() method in JavaScript is used to add one or more elements to the end of an array and return the new length of the array. The push() method modifies the original array, so the elements are added to the end of the original array.

Here's an example of using the push() method:

```
let fruits = ['apple', 'banana'];
let length = fruits.push('cherry');
console.log(fruits); // Output: ['apple', 'banana', 'cherry']
console.log(length); // Output: 3
```

In the example above, the push() method adds the element 'cherry' to the end of the fruits array, and the new length of the fruits array is returned, which is 3.

It's important to note that the push() method modifies the original array. If you need to keep the original array intact, you should make a copy of it before using the push() method.

You can also use the push() method to add multiple elements to an array, like this:

```
let fruits = ['apple', 'banana'];
let length = fruits.push('cherry', 'orange');
console.log(fruits); // Output: ['apple', 'banana', 'cherry', 'orange']
console.log(length); // Output: 4
```

In the example above, the push() method adds the elements 'cherry' and 'orange' to the end of the fruits array, and the new length of the fruits array is returned, which is 4.

**shift()**

The shift() method in JavaScript is used to remove the first element from an array and return the removed element. This method changes the length of the array, shifting all other elements to a lower index, to fill the gap left by the removed element.

Here's an example:

```
let fruits = ['apple', 'banana', 'cherry'];
let first = fruits.shift();
console.log(fruits); // Output: ['banana', 'cherry']
console.log(first); // Output: 'apple'
```

In the example above, the shift() method removes the first element ('apple') from the fruits array and returns it. The returned element can then be stored in a variable, as in this example.

It's important to note that the shift() method only removes the first element of an array. If you need to remove elements from other positions in an array, you can use the splice() method.

**unshift()**

The unshift() method in JavaScript is used to add one or more elements to the beginning of an array and return the new length of the array. This method changes the length of the array and shifts all existing elements to a higher index to make room for the new elements.

Here's an example of using the unshift() method:

```
let fruits = ['banana', 'cherry'];
let length = fruits.unshift('apple');
console.log(fruits); // Output: ['apple', 'banana', 'cherry']
console.log(length); // Output: 3
```

In the example above, the unshift() method adds the element 'apple' to the beginning of the fruits array, and the new length of the fruits array is returned, which is 3.

It's important to note that the unshift() method modifies the original array. If you need to keep the original array intact, you should make a copy of it before using the unshift() method.

You can also use the unshift() method to add multiple elements to an array, like this:

```
let fruits = ['banana', 'cherry'];
let length = fruits.unshift('apple', 'orange');
console.log(fruits); // Output: ['apple', 'orange', 'banana', 'cherry']
console.log(length); // Output: 4
```

In the example above, the unshift() method adds the elements 'apple' and 'orange' to the beginning of the fruits array, and the new length of the fruits array is returned, which is 4.

**Concat():**

The concat method in JavaScript is used to concatenate two or more arrays into a single array. It returns a new array that consists of the elements from the original array(s) and the elements from one or more additional arrays. The original arrays are not modified.

Here's an example of how to use the concat method:

```
let array1 = [1, 2, 3];
let array2 = [4, 5, 6];
let newArray = array1.concat(array2);
console.log(newArray);
// Output: [1, 2, 3, 4, 5, 6]
```

In this example, the concat method is used to concatenate the array array1 and array2 into a new array called newArray. The resulting array contains all the elements from array1 and array2.

The concat method can also be used to concatenate more than two arrays by passing multiple arrays as arguments:

```
let array1 = [1, 2, 3];
let array2 = [4, 5, 6];
let array3 = [7, 8, 9];
let newArray = array1.concat(array2, array3);
console.log(newArray);
// Output: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In this example, the concat method is used to concatenate three arrays into a single array. The resulting array contains all the elements from array1, array2, and array3.

It's important to note that the concat method does not modify the original arrays, but returns a new array that is the result of the concatenation. If you need to modify the original array, you can assign the result of the concat method back to the original array.

**join():**

In JavaScript, the join() method is used to join all elements of an array into a string. The elements are separated by a specified separator. If no separator is provided, the elements are joined with a comma (,) by default.

Here's an example:

```
let fruits = ['apple', 'banana', 'cherry'];
let result = fruits.join();
console.log(result); // 'apple,banana,cherry'
let result = fruits.join('-');
console.log(result); // 'apple-banana-cherry'
```

Note that the join() method does not modify the original array, it returns a new string that consists of all elements joined together.

**slice():**

The slice() method in JavaScript is used to extract a portion of an array and return a new array. The original array is not modified.

Here's the syntax for using the slice() method:

```
array.slice(start, end);
```

The slice() method takes two optional parameters: start and end.

- **start** is the index at which to begin extraction. If start is negative, it is treated as array.length + start.
- **end** is the index before which to end extraction. The element at this index is not included in the extracted portion. If end is omitted, all elements from start to the end of the array will be included in the extracted portion. If end is negative, it is treated as array.length + end.

Here's an example of using the slice() method:

```
let fruits = ['apple', 'banana', 'cherry', 'orange'];
let citrus = fruits.slice(1, 3);
console.log(fruits); // Output: ['apple', 'banana', 'cherry', 'orange']
console.log(citrus); // Output: ['banana', 'cherry']
```

In the example above, the slice() method is used to extract the portion of the fruits array from index 1 ('banana') to index 3 ('cherry'), exclusive. The extracted portion is stored in the citrus array, which is ['banana', 'cherry'].

It's important to note that the slice() method does not modify the original array, but returns a new array that contains the extracted portion.

Here's another example of using the slice() method:

```
let numbers = [1, 2, 3, 4, 5];
let part = numbers.slice(2);
console.log(numbers); // Output: [1, 2, 3, 4, 5]
console.log(part); // Output: [3, 4, 5]
```

In the example above, the slice() method is used to extract the portion of the numbers array from index 2 to the end of the array. The extracted portion is stored in the part array, which is [3, 4, 5].

**splice():**

The splice() method in JavaScript is used to add or remove elements from an array. The method modifies the original array and returns the removed elements (if any) in a new array.

Here's the syntax for using the splice() method:

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

The splice() method takes three mandatory parameters:

- **start:** The index at which to start changing the array. If start is negative, it is treated as array.length + start.
- **deleteCount:** The number of elements to remove. If deleteCount is 0, no elements are removed. If deleteCount is greater than the number of elements that exist after start, all elements from start to the end of the array will be removed.
- **item1, item2, ... :** The elements to add to the array, starting at start. If no elements are added, this parameter can be omitted.

Here's an example of using the splice() method:

```
let numbers = [1, 2, 3, 4, 5];
let removed = numbers.splice(2, 2, 6, 7);
console.log(numbers); // Output: [1, 2, 6, 7, 5]
console.log(removed); // Output: [3, 4]
```

In the example above, the splice() method is used to remove two elements (3 and 4) from the numbers array starting at index 2, and add two elements (6 and 7) in their place. The removed elements are returned in the removed array, which is [3, 4]. The modified numbers array is now [1, 2, 6, 7, 5].

It's important to note that the splice() method can be used to both add and remove elements from an array, and that it modifies the original array. This makes it a powerful tool for manipulating arrays, but it's also important to be careful when using it to ensure that the original array is changed as intended.

**reverse():**

The Array.reverse() method in JavaScript is used to reverse the order of the elements in an array in place. The method modifies the original array and does not create a new array.

Here's an example:

```
let fruits = ['apple', 'banana', 'kiwi', 'mango'];
fruits.reverse();
console.log(fruits);
// Output: ["mango", "kiwi", "banana", "apple"]
```

Note that the reverse method only reverses the order of the elements in the array and does not sort the elements in any way. If you want to sort the elements in ascending or descending order, you should use the Array.sort() method in conjunction with the reverse method.

**indexOf():**

The Array.indexOf() method in JavaScript is used to search the array for an element and return its index. The method returns the first index at which the given element can be found in the array, or -1 if it is not present.

Here's an example:

```
let fruits = ['apple', 'banana', 'kiwi', 'mango'];
let index = fruits.indexOf('kiwi');
console.log(index);
// Output: 2
```

You can also provide a second argument, which specifies the index at which to start searching for the element:

```
let fruits = ['apple', 'banana', 'kiwi', 'mango', 'kiwi'];
let index = fruits.indexOf('kiwi', 3);
console.log(index);
// Output: 4
```

Note that the indexOf method uses the strict equality operator (===) to compare the elements, so if you are searching for a value that is not a string or number, make sure to provide the correct type.