Class-1: Inbuilt Array Methods

1. push()

The `push()` method adds one or more elements to the **end** of an array and returns the new length of the array.

Syntax:

```
javascriptCopy code
array.push(item1, item2, ..., itemX);
```

Parameters:

Parameter	Description
item1, item2, ..., itemX	The items to add to the array. At least one item is required.

Example:

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

Use Case: Adding new items to a shopping cart.

Student Activity:

`Code pen link`

`https://codepen.io/vchandu111/pen/ExBZVQg?editors=1010`

Combine two arrays, one with numbers and one with strings, into a single array using only the `push()` method.

Example:

```
javascriptCopy code
// Define the arrays
let numbers = [1, 2, 3, 4, 5];
let strings = ["one", "two", "three", "four", "five"];

// Combine arrays
for (let i = 0; i < strings.length; i++) {
    numbers.push(strings[i]);
}

console.log(numbers); // Output: [1, 2, 3, 4, 5, "one", "tw
o", "three", "four", "five"]
```

2. pop()

The `pop()` method removes the last element from an array and returns that element. It also changes the length of the array.

Syntax:

```
javascriptCopy code
array.pop();
```

Parameters: The `pop()` method does not take any parameters.

Example:

```
javascriptCopy code
let fruits = ['apple', 'banana', 'orange'];
let lastFruit = fruits.pop();
console.log(lastFruit); // Output: 'orange'
console.log(fruits);    // Output: ['apple', 'banana']
```

Use Case: Removing the last item from a shopping cart.

Student Activity:

`Code pen link:`

https://codepen.io/vchandu111/pen/poXRjVo

Manage a queue of up to 5 people. If adding a new person causes the queue to exceed 5 people, remove the last person.

Example:

```
javascriptCopy code
// Define the initial queue and new person
let queue = ["Alice", "Bob", "Charlie", "Diana", "Eve"];
let newPerson = "Frank";

// Add the new person to the queue
queue.push(newPerson);

// Remove the last person if the queue exceeds 5 people
if (queue.length > 5) {
    queue.pop();
}

console.log(queue); // Output: ["Alice", "Bob", "Charlie", "D
iana", "Frank"]
```

3. shift()

The `shift()` method removes the **first** element from an array and returns that element. It also updates the indexes of the remaining elements.

Example:

```
javascriptCopy code
let fruits = ['apple', 'banana', 'orange'];
let removedFruit = fruits.shift();
```

```
console.log(removedFruit); // Output: 'apple'
console.log(fruits); // Output: ['banana', 'orange']
```

Use Case: Removing the first item from a queue.

Student Activity:

`Codepen link:`

https://codepen.io/vchandu111/pen/eYwgpKJ?editors=0010

Manage a queue of up to 5 people. If adding a new person causes the queue to exceed 5 people, remove the oldest person from the front.

Example:

```
javascriptCopy code
// Define the initial queue and new person
let queue = ["Alice", "Bob", "Charlie", "Diana", "Eve"];
let newPerson = "Frank";

// Add the new person to the queue
queue.push(newPerson);

// Remove the oldest person if the queue exceeds 5 people
if (queue.length > 5) {
    queue.shift();
}

console.log(queue); // Output: ["Bob", "Charlie", "Diana", "E
ve", "Frank"]
```

4. unshift()

The `unshift()` method adds one or more elements to the **beginning** of an array and returns the new length of the array. It also updates the indexes of the existing elements.

Example:

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

Use Case: Adding new items to the beginning of a list.

Student Activity:

`Code pen link:`

https://codepen.io/vchandu111/pen/vYqgNrR

Manage a queue of up to 5 people. Add two VIP customers to the front. If the queue exceeds 5 people, remove the oldest ones.

Example:

```
javascriptCopy code
// Define the initial queue and VIP customers
let queue = ["Charlie", "Diana", "Eve", "Frank", "Grace"];
let vipCustomers = ["Alice", "Bob"];

// Add VIP customers to the front of the queue
for (let i = 0; i < vipCustomers.length; i++) {
    queue.unshift(vipCustomers[i]);
}

// Remove the oldest person if the queue exceeds 5 people
while (queue.length > 5) {
    queue.shift();
}

console.log(queue); // Output: ["Alice", "Bob", "Diana", "Ev
```

```
e", "Frank"]
```

5. concat()

The `concat()` method combines two or more arrays and returns a new array.

Example:

```
javascriptCopy code
let fruits = ['apple', 'banana'];
let moreFruits = ['orange', 'grape'];
let allFruits = fruits.concat(moreFruits);
console.log(allFruits); // Output: ['apple', 'banana', 'orang
e', 'grape']
```

Use Case: Merging multiple arrays into one.

Student Activity:

`Codepen link:`

https://codepen.io/vchandu111/pen/yLdgYqy?editors=0010

Manage a company's employee records with a max capacity of 5. Merge two arrays of current employees and new hires. If the merged list exceeds 5, keep only the most recent 5 employees.

Example:

```
javascriptCopy code
// Define the initial lists
let currentEmployees = ["Alice", "Bob", "Charlie"];
let newHires = ["Diana", "Eve", "Frank", "Grace"];

// Combine the lists using concat()
let allEmployees = currentEmployees.concat(newHires);

// Truncate to the last 5 employees if needed
```

```
if (allEmployees.length > 5) {
    allEmployees.length = 5;
}

console.log(allEmployees); // Output: ["Charlie", "Diana", "E
ve", "Frank", "Grace"]
```

indexOf()

The `indexOf()` method returns the **first index** at which a given element can be found in the array. If the element is not found, it returns `-1`. This method is useful for locating the position of an element within an array.

Syntax:

```
javascriptCopy code
array.indexOf(element, start);
```

Parameters:

Parameter	Description
element	The item to search for in the array.
start	Optional. The index at which to start the search. Default is `0`.

Example:

```
javascriptCopy code
let numbers = [10, 20, 30, 40, 50];
let index = numbers.indexOf(30);
console.log(index); // Output: 2

index = numbers.indexOf(60);
```

```
console.log(index); // Output: -1
```

Use Case: Finding the position of a specific number in a list of values, such as locating a particular measurement in a series of data points.

Student Activity:

`Codepen link:`

https://codepen.io/vchandu111/pen/yLdgYRN

You have an array of integer values. Write a function to find the position of a specific number in the array. If the number is not found, return a message saying "Number not found."

Example:

```
javascriptCopy code
// Define the array of numbers
let values = [5, 15, 25, 35, 45];

// Function to find number index
function findNumberIndex(number) {
    let index = values.indexOf(number);
    if (index === -1) {
        return "Number not found";
    } else {
        return `Number found at index ${index}`;
    }
}

// Test the function
console.log(findNumberIndex(25)); // Output: Number found at
index 2
console.log(findNumberIndex(55)); // Output: Number not found
```

includes()

The `includes()` method checks if an array contains a specific element. It returns `true` if the element is found, and `false` otherwise. This method is useful for determining the presence of an item within an array.

Syntax:

```
javascriptCopy code
array.includes(element, start);
```

Parameters:

Parameter	Description
element	The item to check for in the array.
start	Optional. The index at which to start the search. Default is `0`.

Example:

```
javascriptCopy code
let numbers = [10, 20, 30, 40, 50];
let hasThirty = numbers.includes(30);
console.log(hasThirty); // Output: true

let hasSixty = numbers.includes(60);
console.log(hasSixty); // Output: false
```

Use Case: Checking if a specific value is present in a list, such as verifying if a user ID exists in a list of registered users.

Student Activity:

`Codepen :`

https://codepen.io/vchandu111/pen/XWLpmyN?editors=1010

You have an array of integers. Write a function to check if a specific number is present in the array. If the number is found, return `true`; otherwise, return `false`.

Example:

```
javascriptCopy code
// Define the array of numbers
let values = [5, 15, 25, 35, 45];

// Function to check if number is included
function containsNumber(number) {
    return values.includes(number);
}

// Test the function
console.log(containsNumber(25)); // Output: true
console.log(containsNumber(55)); // Output: false
```

---

join()

The `join()` method combines all elements of an array into a single string, with each element separated by a specified separator. If no separator is provided, the default separator is a comma (`,`).

Syntax:

```
javascriptCopy code
array.join(separator);
```

Parameters:

Parameter	Description
separator	Optional. A string to separate each element in the resulting string. If omitted, the default is a comma.

Example:

```
javascriptCopy code
let fruits = ['apple', 'banana', 'orange'];
let fruitString = fruits.join(', ');
console.log(fruitString); // Output: "apple, banana, orange"

let numberString = [1, 2, 3].join('-');
console.log(numberString); // Output: "1-2-3"
```

Use Case: Creating a formatted string from an array of values, such as generating a comma-separated list of names or values for display or export.

Student Activity:

`Codepen:`

https://codepen.io/vchandu111/pen/WNqRQYV

You have an array with date components (year, month, day) and need to combine them into a formatted date string.

```
let dateComponents = ["2024", "7", "29"];

// Function to format date components
function formatDate(components) {
    // Combine the components into a formatted date string using
    return components.join('-');
}

// Test the function
let formattedDate = formatDate(dateComponents);
console.log(formattedDate); // Output: "2024-7-29"
```

JavaScript Array slice()

The `slice()` method returns a shallow copy of a portion of an array into a new array object.

slice() Syntax

The syntax of the `slice()` method is:

```
arr.slice(start, end)
```

Here, arr is an array.

slice() Parameters

The `slice()` method takes in:

- start (optional) - Starting index of the selection. If not provided, the selection starts at start **0**.
- end (optional) - Ending index of the selection (exclusive). If not provided, the selection ends at the index of the last element.

slice() Return Value

- Returns a new array containing the extracted elements.

Example 1: JavaScript slice() method

```
let languages = ["JavaScript", "Python", "C", "C++", "Java"];

// slicing the array (from start to end)
let new_arr = languages.slice();
console.log(new_arr); // [ 'JavaScript', 'Python', 'C', 'C+
+', 'Java' ]
```

```
// slicing from the third element
let new_arr1 = languages.slice(2);
console.log(new_arr1); // [ 'C', 'C++', 'Java' ]

// slicing from the second element to fourth element
let new_arr2 = languages.slice(1, 4);
console.log(new_arr2); // [ 'Python', 'C', 'C++' ]
```

Run Code

Output

```
[ 'JavaScript', 'Python', 'C', 'C++', 'Java' ]
[ 'C', 'C++', 'Java' ]
[ 'Python', 'C', 'C++' ]
```

Example 2: JavaScript slice() With Negative index

In JavaScript, you can also use negative **start** and **end** indices. The index of the last element is **-1**, the index of the second last element is **-2**, and so on.

```
const languages = ["JavaScript", "Python", "C", "C++", "Jav
a"];

// slicing the array from start to second-to-last
let new_arr = languages.slice(0, -1);
console.log(new_arr); // [ 'JavaScript', 'Python', 'C', 'C++'
]

// slicing the array from third-to-last
let new_arr1 = languages.slice(-3);
console.log(new_arr1); // [ 'C', 'C++', 'Java' ]
```

Run Code

Output

```
[ 'JavaScript', 'Python', 'C', 'C++' ]
[ 'C', 'C++', 'Java' ]
```

Student activity:

Codepen link

https://codepen.io/vchandu111/pen/LYKxGpm?editors=0010

Question 1

Input Array:

```
javascriptCopy code
let numbers = [1, 2, 3, 4, 5];
```

Solution:

```
javascriptCopy code
let result = numbers.slice(1, 4);
console.log(result); // Expected Output: [2, 3, 4]
```

Question 2

Input Array:

```
javascriptCopy code
let names = ["Alice", "Bob", "Charlie", "Dave"];
```

Solution:

```
javascriptCopy code
let result = names.slice(1, 3);
console.log(result); // Expected Output: ["Bob", "Charlie"]
```

Question 3

Input Array:

```
javascriptCopy code
let letters = ["a", "b", "c", "d", "e"];
```

Solution:

```
javascriptCopy code
let result = letters.slice(2, 4);
console.log(result); // Expected Output: ["c", "d"]
```

Question 4

Input Array:

```
javascriptCopy code
let colors = ["red", "green", "blue", "yellow"];
```

Solution:

```
javascriptCopy code
let result = colors.slice(1, 4);
console.log(result); // Expected Output: ["green", "blue", "y
ellow"]
```

Question 5

Input Array:

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

Solution:

```
javascriptCopy code
let result = numbers.slice(2);
console.log(result); // Expected Output: [30, 40, 50]
```

splice()

The `splice()` method changes the contents of an array by removing, replacing, or adding elements. It takes three parameters: the start index, the number of elements to remove, and optional elements to add. It's like modifying a specific portion of an array.

Example:

```
javascriptCopy code
let fruits = ['apple', 'banana', 'orange', 'grape'];
fruits.splice(1, 2, 'kiwi', 'mango');
console.log(fruits); // Output: ['apple', 'kiwi', 'mango', 'g
rape']
```

Use case: Modifying specific elements within an array.

Activity 11: Modify the To-Do List (splice)Objective: Use the `splice()` method to modify the contents of an array.

Scenario: You have a to-do list but need to replace some tasks with new ones.

Task:

- Start with an array `toDoList = ['task1', 'task2', 'task3', 'task4']`.

- Use the `splice()` method to replace 'task2' and 'task3' with 'newTask1' and 'newTask2'. Print the updated list.

Example:

```
javascriptCopy code
let toDoList = ['task1', 'task2', 'task3', 'task4'];
toDoList.splice(1, 2, 'newTask1', 'newTask2');
console.log(toDoList); // Output: ['task1', 'newTask1', 'newT
ask2', 'task4']
```

These notes, examples, and tasks will help students understand and practice each array method in a hands-on and engaging way, ensuring they grasp the concepts and can apply them in real-world scenarios.

JavaScript Array splice()

The `splice()` method is a built-in method for JavaScript Array objects. It lets you change the content of your array by removing or replacing existing elements with new ones.

This method modifies the original array and returns the removed elements as a new array.

In this tutorial, you will learn how you can remove, add, or replace elements of an array using the `splice()` method. Let's start with removing elements from an array first.

How to remove array elements with splice()

For example, suppose you have an array named `months` but you have some day names in the array as follows:

`let months = ["January", "February", "Monday", "Tuesday"];` A mixed array of month and day names

You can use the `splice()` method to remove the day names from the `months` method and add it to a new array at the same time:

```
let months = ["January", "February", "Monday", "Tuesday"];
let days = months.splice(2);

console.log(days); // ["Monday", "Tuesday"]
```

Creating an array of days

The `splice()` method needs at least one parameter, which is the `start` index where the splice operation starts. In the code above, the number `2` is passed to the method, which means `splice()` will start removing elements from index `2`.

You can also define how many elements you want to remove from the array by passing a second `number` argument known as `removeCount`. For example, to remove only one element, you can pass the number `1` like this:

```
let months = ["January", "February", "Monday", "Tuesday"];
let days = months.splice(2, 1);

console.log(days); // ["Monday"]
console.log(months); // ["January", "February", "Tuesday"]
```

Remove only one element from the array

When you omit the `removeCount` parameter, `splice()` will remove all elements from the `start` index to the end of the array.

How to remove and add array elements with splice()

The method also allows you to add new elements right after the delete operation. You just need to pass the elements you want to add to the array after the delete count.

The full syntax of the `splice()` method is as follows:

```
Array.splice(start, removeCount, newItem, newItem, newItem, ...)
```

Complete array splice() method syntax

The following example shows how you can remove "Monday" and "Tuesday" while adding "March" and "April" to the `months` array:

```
let months = ["January", "February", "Monday", "Tuesday"];
let days = months.splice(2, 2, "March", "April");
```

```
console.log(days); // ["Monday", "Tuesday"]
console.log(months); // ["January", "February", "March", "April"]
```

Using splice() to both remove and add elements to an array

How to add new array elements without removing any elements

Finally, you can add new elements without removing any by passing the number `0` to the `removeCount` parameter. When no elements are removed, the splice method will return an empty array. You can choose whether to store the returned empty array to a variable or not.

The following example shows how you can add a new element `"March"` next to `"February"` without deleting any elements. Since the `splice()` method returns an empty array, you don't need to store the returned array:

```
let months = ["January", "February", "Monday", "Tuesday"];
months.splice(2, 0, "March");

console.log(months);
// ["January", "February", "March", "Monday", "Tuesday"]
```

The splice() method called without returning any elements

Student activity

You are managing a list of students enrolled in a class. Use the splice() method to perform the following operations:

- **Add a Student:** Add a student 'Eve' at position 2:
- **Remove a Student: Remove the student at position 1 (Bob)**
- **Replace a Student: Replace the student at position 0 (Alice) with 'Frank'**

Codepen - https://codepen.io/vchandu111/pen/JjQEJoG?editors=1010

```
let students = ['Alice', 'Bob', 'Charlie', 'David'];

// Add a student 'Eve' at position 2
students.splice(2, 0, 'Eve');
```

```
// students is now ['Alice', 'Bob', 'Eve', 'Charlie', 'David']

// Remove the student at position 1 (Bob)
students.splice(1, 1);
// students is now ['Alice', 'Eve', 'Charlie', 'David']

// Replace the student at position 0 (Alice) with 'Frank'
students.splice(0, 1, 'Frank');
// students is now ['Frank', 'Eve', 'Charlie', 'David']

console.log(students);
```