Working with String and Arrays

Demo - 1

edureka!

Problem Statement:

The task at hand is to educate learners about the diverse methods associated with JavaScript's String object and the array of operations that can be performed on Arrays. In this practical learning session, you'll be delving deep into the world of JavaScript, exploring the multifaceted techniques used in manipulating strings and efficiently operating on array elements. As you progress through this interactive module, you will gain a thorough understanding of both String and Array functionalities. This experience is designed to equip you with the knowledge and skills to adeptly handle these critical aspects of JavaScript, enhancing your coding efficiency and problem-solving capabilities in the process.

Solution:

String Methods in JavaScript: JavaScript provides a vast range of methods to perform different operations on strings. These methods are essential for manipulating and handling text in various applications.

1. charAt(): Returns the character at a specified index.

Example:

```
var str = "Hello World!";
console.log(str.charAt(6)); // Output: "W"
```

2. concat(): Concatenates two or more strings.

Example:

```
var str1 = "Hello";
var str2 = " World!";
console.log(str1.concat(str2)); // Output: "Hello World!"
```

3. includes(): Determines whether a string contains a certain substring.

Example:

```
var str = "Hello World!";
console.log(str.includes("World")); // Output: true
```

4. indexOf(): Returns the index of the first occurrence of a specified value.

Example:

```
var str = "Hello World!";
console.log(str.indexOf("World")); // Output: 6
```

5. replace(): Replaces a specified value with another value in a string.

Example:

```
var str = "Hello World!";
console.log(str.replace("World", "Universe"));
```

6. slice(): Extracts a part of a string and returns it as a new string.

Example:

```
var str = "Hello World!";
console.log(str.slice(6, 11)); // Output: "World"
```

7. split(): Splits a string into an array of substrings.

Example:

```
var str = "Hello World!";
console.log(str.split(" ")); // Output: ["Hello", "World!"]
```

8. toLowerCase(): Converts a string to lowercase letters.

Example:

```
var str = "Hello World!";
console.log(str.toLowerCase()); // Output: "hello world!"
```

9. toUpperCase(): Converts a string to uppercase letters.

Example:

```
var str = "Hello World!";
console.log(str.toUpperCase()); // Output: "HELLO WORLD!"
```

10. trim(): Removes whitespace from both ends of a string.

Example:

```
var str = "    Hello World!    ";
console.log(str.trim()); // Output: "Hello World!"
```

11. substring(): Returns the part of the string between the start and end indexes.

Example:

```
var str = "Hello World!";
console.log(str.substring(0, 5)); // Output: "Hello"
```

12. repeat(): Returns a new string with a specified number of copies of the string it is called on.

Example:

```
var str = "Hello ";
console.log(str.repeat(3)); // Output: "Hello Hello Hello "
```

13. search(): Executes a search for a match between a regular expression and this String object.

Example:

```
var str = "Hello World!";
console.log(str.search("World")); // Output: 6
```

14. match(): Retrieves the matches when matching a string against a regular expression.

Example:

```
var str = "The rain in SPAIN stays mainly in the plain";
console.log(str.match(/ain/g)); // Output: ["ain", "ain", "ain"]
```

Conclusion

String methods in JavaScript are powerful tools that allow developers to manipulate, search, and modify text in various ways. These methods enhance the functionality and flexibility of handling strings in any JavaScript application. Understanding and utilizing these methods can significantly improve the efficiency and effectiveness of text processing and manipulation.

Arrays in JavaScript: They are used to store multiple values in a single variable and offer a wide range of operations, making them a versatile and essential part of programming in JavaScript. Below, we'll explore key array operations with examples and their expected outputs.

1. Adding Elements
2. Removing Elements
3. Finding Elements
4. Iterating Over Elements
5. Transforming Arrays
6. Other Useful Methods

Adding Elements

- **push():** Adds elements to the end of an array.

  Example:

```
let fruits = ["Apple", "Banana"];
fruits.push("Cherry");
```

  Output: ["Apple", "Banana", "Cherry"]

- **unshift ():** Adds elements to the beginning of an array.

  Example:

```
fruits.unshift("Mango");
```

  Output: ["Mango", "Apple", "Banana", "Cherry"]

Removing Elements

- **pop():**Removes the last element from an array.

  Example:

```
fruits.pop();
```

  Output: ["Mango", "Apple", "Banana"]

- **shift ():**Removes the first element from an array.

  Example:

```
fruits.shift();
```

  Output: ["Apple", "Banana"]

Finding Elements

- **indexOf():** Returns the first index at which a specified element can be found.

  Example:

```
let index = fruits.indexOf("Banana");
```

  Output: 1

Iterating Over Elements

- **forEach():** Executes a provided function once for each array element.

  Example:

```
fruits.forEach((fruit) => console.log(fruit));
```

  Output:
  Apple

Banana

Transforming Arrays

- **map():** Creates a new array with the results of calling a function for every array element.

  Example:

```
let lengths = fruits.map((fruit) => fruit.length);
```

  Output: [5, 6]

- **filter():** Creates a new array with all elements that pass the test implemented by the provided function.

  Example:

```
let longFruits = fruits.filter((fruit) => fruit.length > 5);
```

  Output: ["Banana"]

- **reduce():** Reduces the array to a single value by executing a reducer function for each element.

  Example:

```
let numbers = [1, 2, 3, 4];
let sum = numbers.reduce((total, number) => total + number, 0);
```

  Output: 10

Other Useful Methods

- **slice():** Returns a shallow copy of a portion of an array.

  Example:

```
let citrus = fruits.slice(1, 3);
```

  Output: ["Banana"]

- **splice():** Changes the contents of an array by removing or replacing existing elements and/or adding new elements.

  Example:

```
fruits.splice(1, 0, "Strawberry");
```

  Output: ["Apple", "Strawberry", "Banana"]

- **join():** Joins all elements of an array into a string.

  Example:

```
let fruitString = fruits.join(", ");
```

  Output: ["Apple", "Strawberry", "Banana"]

- **sort():** Sorts the elements of an array.

  Example:

```
fruits.sort();
```

  Output: ["Apple", "Banana","Strawberry",]

Conclusion

JavaScript arrays are powerful and flexible, providing numerous methods for managing and manipulating data. Whether you're adding, removing, finding, iterating over, transforming, or performing other operations, understanding

these methods is crucial for effective JavaScript programming. Remember, while some methods modify the array in place, others return a new array, so it's important to use them appropriately based on your needs.