

# String and Array Methods

## Training Material (Part 2 of 9)

---

### 1. STRING METHODS

#### What are String Methods?

String methods are built-in functions that allow you to manipulate and work with text data in JavaScript. Strings are immutable, so these methods return new strings.

#### Why use String Methods?

- Transform text (uppercase, lowercase, trim)
- Search and extract parts of strings
- Replace text
- Split and join strings
- Validate and format user input

#### How to use String Methods?

##### Basic String Methods

```
javascript
```

```

const text = " Hello JavaScript World ";

// Length property
console.log(text.length); // 27

// Trim whitespace
console.log(text.trim()); // "Hello JavaScript World"
console.log(text.trimStart()); // "Hello JavaScript World "
console.log(text.trimEnd()); // " Hello JavaScript World"

// Case conversion
console.log(text.toUpperCase()); // " HELLO JAVASCRIPT WORLD "
console.log(text.toLowerCase()); // " hello javascript world "

// charAt and charCodeAt
const word = "JavaScript";
console.log(word.charAt(0)); // "J"
console.log(word.charCodeAt(0)); // 74 (ASCII code)

```

## Searching Strings

```

javascript

const sentence = "JavaScript is awesome and JavaScript is powerful";

// indexOf - returns first occurrence
console.log(sentence.indexOf("JavaScript")); // 0
console.log(sentence.indexOf("is")); // 11
console.log(sentence.indexOf("Python")); // -1 (not found)

// lastIndexOf - returns last occurrence
console.log(sentence.lastIndexOf("JavaScript")); // 26

// includes - returns boolean
console.log(sentence.includes("awesome")); // true
console.log(sentence.includes("difficult")); // false

// startsWith and endsWith
console.log(sentence.startsWith("Java")); // true
console.log(sentence.endsWith("powerful")); // true
console.log(sentence.startsWith("Script", 4)); // true (starts at index 4)

```

## Extracting Parts of Strings

```
javascript

const text = "JavaScript Programming";

// slice(start, end) - extracts a section
console.log(text.slice(0, 10)); // "JavaScript"
console.log(text.slice(11)); // "Programming"
console.log(text.slice(-11)); // "Programming" (negative = from end)

// substring(start, end) - similar to slice
console.log(text.substring(0, 10)); // "JavaScript"
console.log(text.substring(11, 22)); // "Programming"

// substr(start, length) - deprecated but still used
console.log(text.substr(0, 10)); // "JavaScript"
console.log(text.substr(11, 11)); // "Programming"
```

## Replacing and Splitting

```
javascript

const message = "I love JavaScript. JavaScript is great!";

// replace - replaces first occurrence
console.log(message.replace("JavaScript", "Python"));
// "I love Python. JavaScript is great!"

// replaceAll - replaces all occurrences
console.log(message.replaceAll("JavaScript", "Python"));
// "I love Python. Python is great!"

// split - converts string to array
const words = "apple,banana,orange".split(",");
console.log(words); // ["apple", "banana", "orange"]

const chars = "Hello".split("");
console.log(chars); // ["H", "e", "l", "l", "o"]
```

## Template Literals (String Interpolation)

```
javascript
```

```

const name = "Alice";
const age = 25;
const city = "New York";

// Old way
const oldMessage = "My name is " + name + " and I am " + age + " years old.';

// Modern way with template literals
const newMessage = `My name is ${name} and I am ${age} years old.`;
console.log(newMessage);

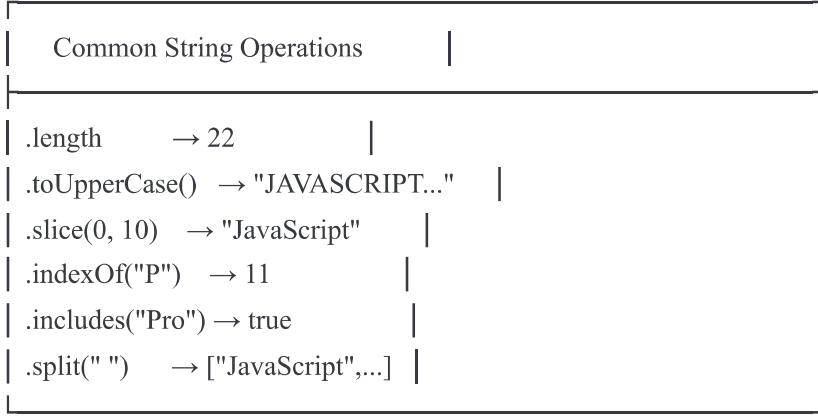
// Multi-line strings
const address =
  Name: ${name}
  Age: ${age}
  City: ${city}
`;
console.log(address);

```

## Visual Diagram: String Methods

String: "JavaScript Programming"

0123456789...



## 2. ARRAY METHODS

### What are Array Methods?

Array methods are built-in functions that allow you to manipulate, transform, and work with array data efficiently.

## Why use Array Methods?

- Add, remove, and modify elements
- Search and filter data
- Transform arrays
- Iterate through elements
- Chain multiple operations

## How to use Array Methods?

### Adding and Removing Elements

javascript

```
const fruits = ["apple", "banana"];

// push - add to end (mutates original)
fruits.push("orange");
console.log(fruits); // ["apple", "banana", "orange"]

// pop - remove from end (mutates original)
const last = fruits.pop();
console.log(last); // "orange"
console.log(fruits); // ["apple", "banana"]

// unshift - add to beginning (mutates original)
fruits.unshift("mango");
console.log(fruits); // ["mango", "apple", "banana"]

// shift - remove from beginning (mutates original)
const first = fruits.shift();
console.log(first); // "mango"
console.log(fruits); // ["apple", "banana"]

// splice - add/remove at any position (mutates original)
const colors = ["red", "green", "blue"];
colors.splice(1, 1, "yellow", "purple"); // Remove 1 at index 1, add 2
console.log(colors); // ["red", "yellow", "purple", "blue"]
```

### Searching and Finding

javascript

```

const numbers = [10, 20, 30, 40, 50];

// indexOf - returns index or -1
console.log(numbers.indexOf(30)); // 2
console.log(numbers.indexOf(60)); // -1

// includes - returns boolean
console.log(numbers.includes(40)); // true
console.log(numbers.includes(60)); // false

// find - returns first matching element
const users = [
  { id: 1, name: "Alice", age: 25 },
  { id: 2, name: "Bob", age: 30 },
  { id: 3, name: "Charlie", age: 25 }
];

const user = users.find(u => u.age === 25);
console.log(user); // { id: 1, name: "Alice", age: 25 }

// findIndex - returns index of first match
const index = users.findIndex(u => u.name === "Bob");
console.log(index); // 1

// some - returns true if ANY element matches
const hasAdult = users.some(u => u.age >= 18);
console.log(hasAdult); // true

// every - returns true if ALL elements match
const allAdults = users.every(u => u.age >= 18);
console.log(allAdults); // true

```

## Transforming Arrays

javascript

```

const numbers = [1, 2, 3, 4, 5];

// map - transforms each element
const doubled = numbers.map(n => n * 2);
console.log(doubled); // [2, 4, 6, 8, 10]

const users = [
  { firstName: "Alice", lastName: "Smith" },
  { firstName: "Bob", lastName: "Jones" }
];
const fullNames = users.map(u => `${u.firstName} ${u.lastName}`);
console.log(fullNames); // ["Alice Smith", "Bob Jones"]

// filter - keeps elements that match condition
const evenNumbers = numbers.filter(n => n % 2 === 0);
console.log(evenNumbers); // [2, 4]

const adults = [
  { name: "Alice", age: 25 },
  { name: "Bob", age: 17 },
  { name: "Charlie", age: 30 }
].filter(person => person.age >= 18);
console.log(adults); // [{ name: "Alice", age: 25 }, { name: "Charlie", age: 30 }]

// reduce - reduces array to single value
const sum = numbers.reduce((total, num) => total + num, 0);
console.log(sum); // 15

const products = [
  { name: "Laptop", price: 1000 },
  { name: "Phone", price: 500 },
  { name: "Tablet", price: 300 }
];
const totalPrice = products.reduce((sum, product) => sum + product.price, 0);
console.log(totalPrice); // 1800

```

## Sorting and Reversing

javascript

```

const fruits = ["banana", "apple", "orange", "mango"];

// sort - sorts in place (mutates original)
fruits.sort();
console.log(fruits); // ["apple", "banana", "mango", "orange"]

// Sorting numbers (need compare function)
const numbers = [40, 100, 1, 5, 25, 10];
numbers.sort((a, b) => a - b); // Ascending
console.log(numbers); // [1, 5, 10, 25, 40, 100]

numbers.sort((a, b) => b - a); // Descending
console.log(numbers); // [100, 40, 25, 10, 5, 1]

// Sorting objects
const users = [
  { name: "Charlie", age: 30 },
  { name: "Alice", age: 25 },
  { name: "Bob", age: 35 }
];
users.sort((a, b) => a.age - b.age);
console.log(users);
// [{name: "Alice", age: 25}, {name: "Charlie", age: 30}, {name: "Bob", age: 35}]

// reverse - reverses array in place (mutates original)
const letters = ["a", "b", "c", "d"];
letters.reverse();
console.log(letters); // ["d", "c", "b", "a"]

```

## Joining and Slicing

javascript

```

// join - converts array to string
const words = ["Hello", "JavaScript", "World"];
console.log(words.join(" ")); // "Hello JavaScript World"
console.log(words.join("-")); // "Hello-JavaScript-World"
console.log(words.join(""))); // "HelloJavaScriptWorld"

// slice - extracts a section (doesn't mutate)
const numbers = [1, 2, 3, 4, 5];
const subset = numbers.slice(1, 4);
console.log(subset); // [2, 3, 4]
console.log(numbers); // [1, 2, 3, 4, 5] (original unchanged)

// concat - combines arrays (doesn't mutate)
const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
const combined = arr1.concat(arr2);
console.log(combined); // [1, 2, 3, 4, 5, 6]

// Spread operator (modern alternative)
const modern = [...arr1, ...arr2];
console.log(modern); // [1, 2, 3, 4, 5, 6]

```

## Array Iteration Methods

```

javascript

const numbers = [1, 2, 3, 4, 5];

// forEach - executes function for each element
numbers.forEach((num, index) => {
  console.log(`Index ${index}: ${num}`);
});

// map vs forEach
// map returns new array, forEach doesn't
const doubled = numbers.map(n => n * 2); // Returns [2, 4, 6, 8, 10]
numbers.forEach(n => n * 2); // Returns undefined

```

## Visual Diagram: Array Method Categories

## ARRAY METHODS OVERVIEW

### MUTATING METHODS (Change original array)

- └─ push, pop, shift, unshift
- └─ splice, sort, reverse
- └─ fill

### NON-MUTATING (Return new array/value)

- └─ slice, concat
- └─ map, filter, reduce
- └─ find, findIndex
- └─ join

### BOOLEAN METHODS (Return true/false)

- └─ includes, some, every
- └─ Array.isArray()

## COMPLETE EXAMPLE: E-Commerce Product Filter

html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Product Filter Demo</title>
<style>
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    min-height: 100vh;
    padding: 20px;
}
.container {
    max-width: 1200px;
    margin: 0 auto;
    background: white;
    border-radius: 15px;
    padding: 30px;
    box-shadow: 0 20px 60px rgba(0,0,0,0.3);
}
h1 {
    color: #333;
    margin-bottom: 30px;
    text-align: center;
}
.controls {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 15px;
    margin-bottom: 30px;
    padding: 20px;
    background: #f8f9fa;
    border-radius: 10px;
}
input, select, button {
    padding: 12px;
    border: 2px solid #ddd;
```

```
border-radius: 8px;
font-size: 14px;
}

input:focus, select:focus {
outline: none;
border-color: #667eea;
}

button {
background: #667eea;
color: white;
border: none;
cursor: pointer;
font-weight: bold;
transition: background 0.3s;
}

button:hover {
background: #5568d3;
}

.products {
display: grid;
grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
gap: 20px;
margin-top: 20px;
}

.product-card {
border: 2px solid #e0e0e0;
border-radius: 10px;
padding: 20px;
background: white;
transition: all 0.3s;
}

.product-card:hover {
transform: translateY(-5px);
box-shadow: 0 10px 25px rgba(0,0,0,0.1);
border-color: #667eea;
}

.product-name {
font-size: 18px;
font-weight: bold;
color: #333;
margin-bottom: 10px;
}

.product-category {
display: inline-block;
```

```
background: #667eea;
color: white;
padding: 4px 12px;
border-radius: 15px;
font-size: 12px;
margin-bottom: 10px;
}

.product-price {
font-size: 24px;
color: #667eea;
font-weight: bold;
margin: 10px 0;
}

.product-rating {
color: #ffd700;
font-size: 18px;
}

.stats {
background: #f8f9fa;
padding: 15px;
border-radius: 8px;
margin-bottom: 20px;
display: flex;
justify-content: space-around;
flex-wrap: wrap;
gap: 10px;
}

.stat-item {
text-align: center;
}

.stat-value {
font-size: 24px;
font-weight: bold;
color: #667eea;
}

.stat-label {
font-size: 12px;
color: #666;
margin-top: 5px;
}

</style>
</head>
<body>
<div class="container">
```

```
<h1>🛒 E-Commerce Product Filter</h1>

<div class="stats" id="stats"></div>

<div class="controls">
  <input type="text" id="searchInput" placeholder="Search products...">
  <select id="categoryFilter">
    <option value="all">All Categories</option>
    <option value="electronics">Electronics</option>
    <option value="clothing">Clothing</option>
    <option value="books">Books</option>
  </select>
  <select id="sortBy">
    <option value="name">Sort by Name</option>
    <option value="price-low">Price: Low to High</option>
    <option value="price-high">Price: High to Low</option>
    <option value="rating">Highest Rating</option>
  </select>
  <button onclick="resetFilters()">Reset Filters</button>
</div>
```

```
<div class="products" id="productsContainer"></div>
</div>
```

```
<script>
  // Sample product data
  const products = [
    { id: 1, name: "Wireless Headphones", category: "electronics", price: 79.99, rating: 4.5 },
    { id: 2, name: "JavaScript Book", category: "books", price: 29.99, rating: 4.8 },
    { id: 3, name: "Cotton T-Shirt", category: "clothing", price: 19.99, rating: 4.2 },
    { id: 4, name: "Smart Watch", category: "electronics", price: 199.99, rating: 4.6 },
    { id: 5, name: "Python Programming", category: "books", price: 34.99, rating: 4.7 },
    { id: 6, name: "Denim Jeans", category: "clothing", price: 49.99, rating: 4.3 },
    { id: 7, name: "Laptop Stand", category: "electronics", price: 39.99, rating: 4.4 },
    { id: 8, name: "Web Design Guide", category: "books", price: 24.99, rating: 4.5 },
    { id: 9, name: "Hoodie", category: "clothing", price: 45.99, rating: 4.6 },
    { id: 10, name: "USB-C Cable", category: "electronics", price: 12.99, rating: 4.1 }
  ];

```

```
// Display products
function displayProducts(productsToShow) {
  const container = document.getElementById('productsContainer');

  // Use map to create HTML for each product

```

```

const html = productsToShow.map(product => `
  <div class="product-card">
    <div class="product-category">${product.category.toUpperCase()}</div>
    <div class="product-name">${product.name}</div>
    <div class="product-price">${product.price.toFixed(2)}</div>
    <div class="product-rating">${'★'.repeat(Math.floor(product.rating))} ${product.rating}</div>
  </div>
`);

container.innerHTML = html;
updateStats(productsToShow);
}

// Update statistics
function updateStats(productsToShow) {
  const stats = document.getElementById('stats');

  // Use reduce to calculate total and average
  const totalPrice = productsToShow.reduce((sum, p) => sum + p.price, 0);
  const avgPrice = productsToShow.length > 0 ? totalPrice / productsToShow.length : 0;
  const avgRating = productsToShow.length > 0
    ? productsToShow.reduce((sum, p) => sum + p.rating, 0) / productsToShow.length
    : 0;

  stats.innerHTML = `
    <div class="stat-item">
      <div class="stat-value">${productsToShow.length}</div>
      <div class="stat-label">Products</div>
    </div>
    <div class="stat-item">
      <div class="stat-value">${avgPrice.toFixed(2)}</div>
      <div class="stat-label">Avg Price</div>
    </div>
    <div class="stat-item">
      <div class="stat-value">${avgRating.toFixed(1)} ★</div>
      <div class="stat-label">Avg Rating</div>
    </div>
  `;
}

// Filter and sort products
function filterAndSort() {
  const searchTerm = document.getElementById('searchInput').value.toLowerCase();
  const category = document.getElementById('categoryFilter').value;
}

```

```
const sortBy = document.getElementById('sortBy').value;

// Filter products using filter method
let filtered = products.filter(product => {
  // Check if product name includes search term
  const matchesSearch = product.name.toLowerCase().includes(searchTerm);

  // Check if product matches category (or "all" is selected)
  const matchesCategory = category === 'all' || product.category === category;

  return matchesSearch && matchesCategory;
});

// Sort products using sort method
const sorted = [...filtered].sort((a, b) => {
  switch(sortBy) {
    case 'name':
      return a.name.localeCompare(b.name);
    case 'price-low':
      return a.price - b.price;
    case 'price-high':
      return b.price - a.price;
    case 'rating':
      return b.rating - a.rating;
    default:
      return 0;
  }
});

displayProducts(sorted);
}

// Reset all filters
function resetFilters() {
  document.getElementById('searchInput').value = '';
  document.getElementById('categoryFilter').value = 'all';
  document.getElementById('sortBy').value = 'name';
  filterAndSort();
}

// Add event listeners
document.getElementById('searchInput').addEventListener('input', filterAndSort);
document.getElementById('categoryFilter').addEventListener('change', filterAndSort);
document.getElementById('sortBy').addEventListener('change', filterAndSort);
```

```
// Initial display
displayProducts(products);
</script>
</body>
</html>
```

## METHOD CHAINING EXAMPLE

javascript

```
// Powerful combination of array methods
const users = [
  { name: "Alice", age: 25, city: "New York", salary: 50000 },
  { name: "Bob", age: 30, city: "London", salary: 60000 },
  { name: "Charlie", age: 35, city: "New York", salary: 75000 },
  { name: "Diana", age: 28, city: "Paris", salary: 55000 },
  { name: "Eve", age: 32, city: "New York", salary: 70000 }
];

// Find average salary of New York employees over 26
const avgSalary = users
  .filter(user => user.city === "New York") // Filter by city
  .filter(user => user.age > 26)           // Filter by age
  .map(user => user.salary)               // Extract salaries
  .reduce((sum, salary) => sum + salary, 0) // Sum salaries
  / users.filter(u => u.city === "New York" && u.age > 26).length; // Divide by count

console.log(`Average: $$ {avgSalary}`); // Average: $72500

// Get names of top 2 earners in uppercase
const topEarners = users
  .sort((a, b) => b.salary - a.salary) // Sort by salary (descending)
  .slice(0, 2)                      // Take top 2
  .map(user => user.name.toUpperCase()) // Convert to uppercase
  .join(" and ");                  // Join with " and "

console.log(topEarners); // "CHARLIE and EVE"
```

# COMPARISON: STRING VS ARRAY METHODS

Operation	String Method	Array Method
Search	indexOf, includes, search	indexOf, includes, find
Extract	slice, substring, substr	slice, filter
Transform	toUpperCase, toLowerCase, replace	map, filter, reduce
Split/Join	split → array	join → string
Check existence	includes	includes, some
Combine	concat, + operator	concat, spread (...)

## PRACTICE EXERCISES

### Exercise 1: String Manipulation

Write a function that takes a sentence and returns it with the first letter of each word capitalized.

### Exercise 2: Array Filtering

Given an array of numbers, filter out negative numbers, double the remaining ones, and return the sum.

### Exercise 3: Data Transformation

Transform an array of user objects to extract only names and emails into a new format.

### Exercise 4: Complex Filtering

Create a product search that filters by name, price range, and category simultaneously.

## KEY TAKEAWAYS

1. **String methods** are immutable - they always return new strings
2. **Array methods** can be mutating (push, pop) or non-mutating (map, filter)
3. **Method chaining** allows powerful data transformations in one line
4. **Choose the right method:** map for transformation, filter for selection, reduce for aggregation

Next Topic: DOM Manipulation