<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

// Sample data

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

const fruits = ['apple', 'banana', 'cherry'];

const users = [

  { id: 1, name: 'Alice' },

  { id: 2, name: 'Bob' }

];

const mixed = [1, 'hello', true, null, undefined];

const prices = [10, 20, 30];

// 1. Basic iteration and printing each number

console.log("Example 1: Print each number:");

numbers.forEach((num) => {

  console.log(num);

});

// 2. Sum all elements

let sum = 0;

numbers.forEach((num) => {

  sum += num;

});

console.log("\nExample 2: Sum of numbers =", sum);

// 3. Log index and value from fruits array

console.log("\nExample 3: Fruits with index:");

fruits.forEach((fruit, index) => {

  console.log(`Index ${index}: ${fruit}`);

});

// 4. Accessing properties of objects in array

console.log("\nExample 4: User details:");

users.forEach((user) => {

  console.log(`ID: ${user.id}, Name: ${user.name}`);

});

// 5. Modify original array (apply discount)

console.log("\nExample 5: Apply 10% discount on prices:");

prices.forEach((price, index, arr) => {

  arr[index] = price \* 0.9;

});

console.log("Discounted Prices:", prices);

===========================================================

Example 1: Print each number:

1

2

3

4

5

Example 2: Sum of numbers = 15

Example 3: Fruits with index:

Index 0: apple

Index 1: banana

Index 2: cherry

Example 4: User details:

ID: 1, Name: Alice

ID: 2, Name: Bob

Example 5: Apply 10% discount on prices:

Discounted Prices: [ 9, 18, 27 ]

========================================================

// Sample data

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

const names = ['alice', 'bob', 'charlie'];

const users = [

  { id: 1, name: 'Alice', age: 25 },

  { id: 2, name: 'Bob', age: 30 }

];

const strings = ['10', '20', '30'];

const products = [

  { name: 'Pen', price: 10 },

  { name: 'Notebook', price: 50 }

];

// 1. Double each number

const doubled = numbers.map(num => num \* 2);

console.log("Example 1: Doubled numbers:", doubled);

// 2. Capitalize names

const capitalizedNames = names.map(name => name.charAt(0).toUpperCase() + name.slice(1));

console.log("\nExample 2: Capitalized Names:", capitalizedNames);

// 3. Extract only user names from objects

const userNames = users.map(user => user.name);

console.log("\nExample 3: User Names:", userNames);

// 4. Convert string numbers to actual numbers

const numericValues = strings.map(str => parseInt(str));

console.log("\nExample 4: Converted Strings to Numbers:", numericValues);

// 5. Add tax to product prices (e.g., 10% GST)

const productsWithTax = products.map(product => {

  return {

    ...product,

    priceWithTax: product.price \* 1.10

  };

});

console.log("\nExample 5: Products with tax added:");

console.log(productsWithTax);

======================================================

Example 1: Doubled numbers: [2, 4, 6, 8, 10]

Example 2: Capitalized Names: ['Alice', 'Bob', 'Charlie']

Example 3: User Names: ['Alice', 'Bob']

Example 4: Converted Strings to Numbers: [10, 20, 30]

Example 5: Products with tax added:

[

  { name: 'Pen', price: 10, priceWithTax: 11 },

  { name: 'Notebook', price: 50, priceWithTax: 55 }

]

========================================================

// Sample Data

const numbers = [10, 5, 20, 1, 100];

const words = ['banana', 'apple', 'cherry', 'date'];

const users = [

  { name: 'Alice', age: 30 },

  { name: 'Bob', age: 25 },

  { name: 'Charlie', age: 35 }

];

const mixedCase = ['Banana', 'apple', 'Cherry', 'date'];

const prices = [50.5, 10.99, 100.25, 20.75];

// 1. Sort numbers in ascending order

const ascendingNumbers = [...numbers].sort((a, b) => a - b);

console.log("Example 1: Numbers Ascending:", ascendingNumbers);

// 2. Sort strings alphabetically

const sortedWords = [...words].sort(); // Default sort is lexicographic

console.log("\nExample 2: Sorted Words:", sortedWords);

// 3. Sort users by age

const usersByAge = [...users].sort((a, b) => a.age - b.age);

console.log("\nExample 3: Users Sorted by Age:");

console.log(usersByAge);

// 4. Case-insensitive string sort

const caseInsensitive = [...mixedCase].sort((a, b) =>

  a.toLowerCase().localeCompare(b.toLowerCase())

);

console.log("\nExample 4: Case-insensitive Sorted Strings:", caseInsensitive);

// 5. Sort prices in descending order

const pricesDesc = [...prices].sort((a, b) => b - a);

console.log("\nExample 5: Prices Descending:", pricesDesc);

================================================

Example 1: Numbers Ascending: [1, 5, 10, 20, 100]

Example 2: Sorted Words: ['apple', 'banana', 'cherry', 'date']

Example 3: Users Sorted by Age:

[

  { name: 'Bob', age: 25 },

  { name: 'Alice', age: 30 },

  { name: 'Charlie', age: 35 }

]

Example 4: Case-insensitive Sorted Strings: ['apple', 'Banana', 'Cherry', 'date']

Example 5: Prices Descending: [100.25, 50.5, 20.75, 10.99]

=============================================

// Original arrays

const numbers = [4, 2, 7, 1, 9];

const fruits = ['banana', 'Apple', 'cherry', 'Date'];

const users = [

  { name: 'Zara', age: 28 },

  { name: 'Mike', age: 21 },

  { name: 'Anna', age: 35 }

];

const prices = [49.99, 5.5, 19.95, 100.0];

const codes = ['b1', 'a2', 'A1', 'B2'];

// 1. Sort numbers ascending

const sortedNumbers = numbers.toSorted((a, b) => a - b);

console.log("Example 1: Sorted Numbers Ascending:", sortedNumbers);

// 2. Case-insensitive sort of strings

const sortedFruits = fruits.toSorted((a, b) => a.toLowerCase().localeCompare(b.toLowerCase()));

console.log("\nExample 2: Case-insensitive Sorted Fruits:", sortedFruits);

// 3. Sort users by age

const sortedUsers = users.toSorted((a, b) => a.age - b.age);

console.log("\nExample 3: Users Sorted by Age:");

console.log(sortedUsers);

// 4. Sort prices descending

const sortedPrices = prices.toSorted((a, b) => b - a);

console.log("\nExample 4: Prices Sorted Descending:", sortedPrices);

// 5. Alphanumeric sort (case-sensitive)

const sortedCodes = codes.toSorted(); // default lexicographical sort

console.log("\nExample 5: Alphanumeric Sorted Codes:", sortedCodes);

// Check original arrays are not mutated

console.log("\nOriginal Numbers Array:", numbers);

console.log("Original Fruits Array:", fruits);

==================================================================

Example 1: Sorted Numbers Ascending: [1, 2, 4, 7, 9]

Example 2: Case-insensitive Sorted Fruits: ['Apple', 'banana', 'cherry', 'Date']

Example 3: Users Sorted by Age:

[

  { name: 'Mike', age: 21 },

  { name: 'Zara', age: 28 },

  { name: 'Anna', age: 35 }

]

Example 4: Prices Sorted Descending: [100, 49.99, 19.95, 5.5]

Example 5: Alphanumeric Sorted Codes: ['A1', 'B2', 'a2', 'b1']

Original Numbers Array: [4, 2, 7, 1, 9]

Original Fruits Array: ['banana', 'Apple', 'cherry', 'Date']

======================================================

// Sample Data

const numbers = [2, 4, 6, 8];

const names = ['Alice', 'Bob', 'Charlie'];

const mixed = [1, 'hello', true];

const users = [

  { name: 'John', active: true },

  { name: 'Jane', active: true },

  { name: 'Jim', active: true }

];

const ages = [22, 30, 45, 17];

// 1. Check if all numbers are even

const allEven = numbers.every(num => num % 2 === 0);

console.log("Example 1: All numbers are even?", allEven);

// 2. Check if all names are strings

const allStrings = names.every(name => typeof name === 'string');

console.log("\nExample 2: All names are strings?", allStrings);

// 3. Check if all elements are of the same type

const allSameType = mixed.every(val => typeof val === typeof mixed[0]);

console.log("\nExample 3: All elements are same type?", allSameType);

// 4. Check if all users are active

const allActiveUsers = users.every(user => user.active);

console.log("\nExample 4: All users are active?", allActiveUsers);

// 5. Check if all ages are 18 or above

const allAdults = ages.every(age => age >= 18);

console.log("\nExample 5: All are adults (18+)?", allAdults);

=================================================

Example 1: All numbers are even? true

Example 2: All names are strings? true

Example 3: All elements are same type? false

Example 4: All users are active? true

Example 5: All are adults (18+)? false

======================================

// Sample data

const numbers = [1, 3, 5, 7, 10];

const names = ['Alice', 'Bob', 'Charlie'];

const values = [null, undefined, '', false];

const users = [

  { name: 'John', active: false },

  { name: 'Jane', active: true },

  { name: 'Jim', active: false }

];

const words = ['dog', 'cat', 'elephant', 'ant'];

// 1. Check if there is at least one even number

const hasEven = numbers.some(num => num % 2 === 0);

console.log("Example 1: Has even number?", hasEven);

// 2. Check if any name starts with 'A'

const startsWithA = names.some(name => name.startsWith('A'));

console.log("\nExample 2: Any name starts with 'A'?", startsWithA);

// 3. Check if any value is truthy

const hasTruthy = values.some(val => Boolean(val));

console.log("\nExample 3: Any truthy value?", hasTruthy);

// 4. Check if any user is active

const anyActiveUser = users.some(user => user.active);

console.log("\nExample 4: Any active user?", anyActiveUser);

// 5. Check if any word is shorter than 3 characters

const shortWordExists = words.some(word => word.length < 3);

console.log("\nExample 5: Any word shorter than 3 characters?", shortWordExists);

===============================================

Example 1: Has even number? true

Example 2: Any name starts with 'A'? true

Example 3: Any truthy value? false

Example 4: Any active user? true

Example 5: Any word shorter than 3 characters? false

===========================================

// Sample data

const numbers = [1, 2, 3, 4, 5, 6, 7, 8];

const fruits = ['apple', 'banana', 'mango', 'kiwi', 'blueberry'];

const users = [

  { name: 'Alice', age: 25 },

  { name: 'Bob', age: 17 },

  { name: 'Charlie', age: 19 }

];

const prices = [49.99, 10.5, 100, 25.25];

const mixedValues = [0, 'hello', false, null, 42, '', true];

// 1. Filter even numbers

const evenNumbers = numbers.filter(num => num % 2 === 0);

console.log("Example 1: Even Numbers:", evenNumbers);

// 2. Filter fruits that start with 'b'

const fruitsStartingWithB = fruits.filter(fruit => fruit.startsWith('b'));

console.log("\nExample 2: Fruits starting with 'b':", fruitsStartingWithB);

// 3. Filter users who are 18 or older

const adultUsers = users.filter(user => user.age >= 18);

console.log("\nExample 3: Adult Users (18+):", adultUsers);

// 4. Filter prices greater than 30

const expensiveItems = prices.filter(price => price > 30);

console.log("\nExample 4: Prices > 30:", expensiveItems);

// 5. Filter truthy values from mixed array

const truthyValues = mixedValues.filter(Boolean);

console.log("\nExample 5: Truthy Values:", truthyValues);

===================================

Example 1: Even Numbers: [2, 4, 6, 8]

Example 2: Fruits starting with 'b': ['banana', 'blueberry']

Example 3: Adult Users (18+):

[

  { name: 'Alice', age: 25 },

  { name: 'Charlie', age: 19 }

]

Example 4: Prices > 30: [49.99, 100]

Example 5: Truthy Values: ['hello', 42, true]

============================================

// Sample data

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

const words = ['hello', 'world', 'javascript'];

const prices = [10, 20, 30];

const votes = ['yes', 'no', 'yes', 'yes', 'no'];

const products = [

  { name: 'Pen', price: 10, quantity: 2 },

  { name: 'Notebook', price: 20, quantity: 1 },

  { name: 'Pencil', price: 5, quantity: 5 }

];

// 1. Sum of all numbers

const sum = numbers.reduce((acc, curr) => acc + curr, 0);

console.log("Example 1: Sum of numbers:", sum);

// 2. Concatenate all words into a sentence

const sentence = words.reduce((acc, word) => acc + ' ' + word);

console.log("\nExample 2: Sentence from words:", sentence);

// 3. Total price from array of prices

const totalPrice = prices.reduce((total, price) => total + price, 0);

console.log("\nExample 3: Total Price:", totalPrice);

// 4. Count occurrences of each vote

const voteCount = votes.reduce((acc, vote) => {

  acc[vote] = (acc[vote] || 0) + 1;

  return acc;

}, {});

console.log("\nExample 4: Vote Count:", voteCount);

// 5. Calculate total bill from array of product objects

const totalBill = products.reduce((acc, item) => acc + (item.price \* item.quantity), 0);

console.log("\nExample 5: Total Bill:", totalBill);

==========================================

Example 1: Sum of numbers: 15

Example 2: Sentence from words: hello world javascript

Example 3: Total Price: 60

Example 4: Vote Count: { yes: 3, no: 2 }

Example 5: Total Bill: 55

========================================

// Sample data

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

const words = ['one', 'two', 'three'];

const bools = [true, true, false];

const operations = ['multiply', 'add', 'subtract'];

const filePathParts = ['home', 'user', 'documents', 'file.txt'];

// 1. Sum numbers from right to left

const sumRight = numbers.reduceRight((acc, curr) => acc + curr, 0);

console.log("Example 1: Sum from right to left:", sumRight);

// 2. Reverse concatenate words

const reversedSentence = words.reduceRight((acc, word) => acc + ' ' + word);

console.log("\nExample 2: Reverse concatenated sentence:", reversedSentence.trim());

// 3. Logical AND from right to left

const logicalAnd = bools.reduceRight((acc, val) => acc && val, true);

console.log("\nExample 3: Logical AND from right to left:", logicalAnd);

// 4. Simulate applying operations from right to left (e.g., in RPN calculators)

const operationChain = operations.reduceRight((acc, op) => acc + ' → ' + op);

console.log("\nExample 4: Operation sequence from right to left:", operationChain);

// 5. Build full file path (right to left for demonstration)

const filePath = filePathParts.reduceRight((acc, part) => part + '/' + acc);

console.log("\nExample 5: File path built from right to left:", filePath);

===================================================

Example 1: Sum from right to left: 10

Example 2: Reverse concatenated sentence: three two one

Example 3: Logical AND from right to left: false

Example 4: Operation sequence from right to left: subtract → add → multiply

Example 5: File path built from right to left: home/user/documents/file.txt

===========================================

// Example 1: Remove the last number from an array

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

const lastNumber = numbers.pop();

console.log("Example 1: Removed last number:", lastNumber);

console.log("Updated numbers array:", numbers);

// Example 2: Remove the last fruit from the list

const fruits = ['apple', 'banana', 'cherry'];

const lastFruit = fruits.pop();

console.log("\nExample 2: Removed last fruit:", lastFruit);

console.log("Updated fruits array:", fruits);

// Example 3: Pop from an array of mixed types

const mixed = [1, 'hello', true, null];

const lastItem = mixed.pop();

console.log("\nExample 3: Removed last item:", lastItem);

console.log("Updated mixed array:", mixed);

// Example 4: Use pop inside a loop until array is empty

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

console.log("\nExample 4: Popping all elements using loop:");

while (stack.length > 0) {

  console.log("Popped:", stack.pop());

}

console.log("Final stack:", stack);

// Example 5: Pop object from array of objects

const users = [

  { name: 'Alice' },

  { name: 'Bob' },

  { name: 'Charlie' }

];

const lastUser = users.pop();

console.log("\nExample 5: Removed last user:", lastUser);

console.log("Updated users array:", users);

====================================================

Example 1: Removed last number: 40

Updated numbers array: [10, 20, 30]

Example 2: Removed last fruit: cherry

Updated fruits array: ['apple', 'banana']

Example 3: Removed last item: null

Updated mixed array: [1, 'hello', true]

Example 4: Popping all elements using loop:

Popped: 5

Popped: 4

Popped: 3

Popped: 2

Popped: 1

Final stack: []

Example 5: Removed last user: { name: 'Charlie' }

Updated users array: [ { name: 'Alice' }, { name: 'Bob' } ]

=================================================

// Example 1: Push a number into an array

const numbers = [1, 2, 3];

numbers.push(4);

console.log("Example 1: After pushing 4:", numbers);

// Example 2: Push multiple fruits into the array

const fruits = ['apple'];

fruits.push('banana', 'cherry');

console.log("\nExample 2: After pushing multiple fruits:", fruits);

// Example 3: Push an object into an array of users

const users = [{ name: 'Alice' }];

users.push({ name: 'Bob' });

console.log("\nExample 3: After pushing an object:", users);

// Example 4: Push elements conditionally

const messages = [];

const message = 'Hello, world!';

if (message.length > 0) {

  messages.push(message);

}

console.log("\nExample 4: After conditionally pushing message:", messages);

// Example 5: Push using a loop

const letters = [];

for (let char of ['a', 'b', 'c']) {

  letters.push(char);

}

console.log("\nExample 5: After pushing in a loop:", letters);

==========================================================

Example 1: After pushing 4: [1, 2, 3, 4]

Example 2: After pushing multiple fruits: ['apple', 'banana', 'cherry']

Example 3: After pushing an object: [{ name: 'Alice' }, { name: 'Bob' }]

Example 4: After conditionally pushing message: ['Hello, world!']

Example 5: After pushing in a loop: ['a', 'b', 'c']

======================================================

// Example 1: Remove the first number

const numbers = [10, 20, 30];

const firstNum = numbers.shift();

console.log("Example 1: Removed first number:", firstNum);

console.log("Updated numbers array:", numbers);

// Example 2: Remove the first fruit

const fruits = ['apple', 'banana', 'cherry'];

const firstFruit = fruits.shift();

console.log("\nExample 2: Removed first fruit:", firstFruit);

console.log("Updated fruits array:", fruits);

// Example 3: Shift from an array of strings

const tasks = ['task1', 'task2', 'task3'];

console.log("\nExample 3: Task processing:");

while (tasks.length > 0) {

  console.log("Processing:", tasks.shift());

}

console.log("All tasks processed, final array:", tasks);

// Example 4: Shift from an array of objects

const queue = [

  { id: 1, user: 'Alice' },

  { id: 2, user: 'Bob' }

];

const firstUser = queue.shift();

console.log("\nExample 4: Removed first user:", firstUser);

console.log("Updated queue:", queue);

// Example 5: Shift from a mixed-type array

const mixed = [true, 'hello', 42];

const removed = mixed.shift();

console.log("\nExample 5: Removed first mixed value:", removed);

console.log("Updated mixed array:", mixed);

=============================================

Example 1: Removed first number: 10

Updated numbers array: [20, 30]

Example 2: Removed first fruit: apple

Updated fruits array: ['banana', 'cherry']

Example 3: Task processing:

Processing: task1

Processing: task2

Processing: task3

All tasks processed, final array: []

Example 4: Removed first user: { id: 1, user: 'Alice' }

Updated queue: [{ id: 2, user: 'Bob' }]

Example 5: Removed first mixed value: true

Updated mixed array: ['hello', 42]

===================================

// Example 1: Add number at the beginning

const numbers = [2, 3, 4];

numbers.unshift(1);

console.log("Example 1: After unshift:", numbers);

// Example 2: Add multiple fruits to beginning

const fruits = ['orange'];

fruits.unshift('apple', 'banana');

console.log("\nExample 2: After unshifting multiple fruits:", fruits);

// Example 3: Prepend log entries

const logs = ['Log2', 'Log3'];

logs.unshift('Log1');

console.log("\nExample 3: Logs after unshift:", logs);

// Example 4: Add object to beginning of array

const users = [{ name: 'Bob' }];

users.unshift({ name: 'Alice' });

console.log("\nExample 4: Users array after unshift:", users);

// Example 5: Prepend items using loop

const letters = ['d', 'e'];

const newLetters = ['a', 'b', 'c'];

for (let i = newLetters.length - 1; i >= 0; i--) {

  letters.unshift(newLetters[i]);

}

console.log("\nExample 5: Letters array after loop unshift:", letters);

==============================================

Example 1: After unshift: [1, 2, 3, 4]

Example 2: After unshifting multiple fruits: ['apple', 'banana', 'orange']

Example 3: Logs after unshift: ['Log1', 'Log2', 'Log3']

Example 4: Users array after unshift: [{ name: 'Alice' }, { name: 'Bob' }]

Example 5: Letters array after loop unshift: ['a', 'b', 'c', 'd', 'e']

==============================================

// Sample array for examples

const data = ['a', 'b', 'c', 'd', 'e', 'f', 'g'];

// Example 1: Slice first 3 elements

const firstThree = data.slice(0, 3);

console.log("Example 1: First 3 elements:", firstThree);

// Example 2: Slice from index 2 to the end

const fromTwo = data.slice(2);

console.log("\nExample 2: From index 2 to end:", fromTwo);

// Example 3: Slice the last 2 elements using negative indices

const lastTwo = data.slice(-2);

console.log("\nExample 3: Last 2 elements:", lastTwo);

// Example 4: Slice a middle section (e.g., 2nd to 5th elements)

const midSection = data.slice(1, 5);

console.log("\nExample 4: Middle section (1 to 4 index):", midSection);

// Example 5: Create a shallow copy of the entire array

const fullCopy = data.slice();

console.log("\nExample 5: Full shallow copy:", fullCopy);

=============================================

Example 1: First 3 elements: ['a', 'b', 'c']

Example 2: From index 2 to end: ['c', 'd', 'e', 'f', 'g']

Example 3: Last 2 elements: ['f', 'g']

Example 4: Middle section (1 to 4 index): ['b', 'c', 'd', 'e']

Example 5: Full shallow copy: ['a', 'b', 'c', 'd', 'e', 'f', 'g']

========================================

// Example 1: Remove elements starting from index

const items1 = ['a', 'b', 'c', 'd', 'e'];

const removed1 = items1.splice(2, 2); // remove 2 elements from index 2

console.log("Example 1: Removed:", removed1);

console.log("Updated array:", items1);

// Example 2: Insert elements without removing

const items2 = ['a', 'b', 'e'];

items2.splice(2, 0, 'c', 'd'); // insert at index 2

console.log("\nExample 2: After insertion:", items2);

// Example 3: Replace elements

const items3 = ['apple', 'banana', 'cherry'];

items3.splice(1, 1, 'blueberry'); // replace 'banana' with 'blueberry'

console.log("\nExample 3: After replacement:", items3);

// Example 4: Remove last 2 elements

const items4 = ['x', 'y', 'z', 'w'];

items4.splice(-2, 2); // remove last 2 elements using negative index

console.log("\nExample 4: After removing last 2:", items4);

// Example 5: Clear the entire array

const items5 = ['one', 'two', 'three'];

items5.splice(0, items5.length); // remove all elements

console.log("\nExample 5: Array cleared:", items5);

=============================================

Example 1: Removed: ['c', 'd']

Updated array: ['a', 'b', 'e']

Example 2: After insertion: ['a', 'b', 'c', 'd', 'e']

Example 3: After replacement: ['apple', 'blueberry', 'cherry']

Example 4: After removing last 2: ['x', 'y']

Example 5: Array cleared: []

=====================================

// Ensure your environment supports toSplice()

// Original array

const original = ['a', 'b', 'c', 'd', 'e'];

console.log("Original array (unchanged):", original);

// Example 1: Remove 2 elements starting from index 1

const result1 = original.toSplice(1, 2);

console.log("\nExample 1: Remove 2 from index 1 →", result1);

// Example 2: Insert elements without deleting (at index 2)

const result2 = original.toSplice(2, 0, 'x', 'y');

console.log("\nExample 2: Insert 'x', 'y' at index 2 →", result2);

// Example 3: Replace element at index 3

const result3 = original.toSplice(3, 1, 'z');

console.log("\nExample 3: Replace 1 at index 3 with 'z' →", result3);

// Example 4: Remove last element using negative index

const result4 = original.toSplice(-1, 1);

console.log("\nExample 4: Remove last element →", result4);

// Example 5: Insert at beginning

const result5 = original.toSplice(0, 0, 'start');

console.log("\nExample 5: Insert 'start' at the beginning →", result5);

// Confirm original is untouched

console.log("\nOriginal array still unchanged:", original);

===========================================

Original array (unchanged): [ 'a', 'b', 'c', 'd', 'e' ]

Example 1: Remove 2 from index 1 → [ 'a', 'd', 'e' ]

Example 2: Insert 'x', 'y' at index 2 → [ 'a', 'b', 'x', 'y', 'c', 'd', 'e' ]

Example 3: Replace 1 at index 3 with 'z' → [ 'a', 'b', 'c', 'z', 'e' ]

Example 4: Remove last element → [ 'a', 'b', 'c', 'd' ]

Example 5: Insert 'start' at the beginning → [ 'start', 'a', 'b', 'c', 'd', 'e' ]

Original array still unchanged: [ 'a', 'b', 'c', 'd', 'e' ]

=========================================

// Example 1: Concatenating two arrays of numbers

const arr1 = [1, 2];

const arr2 = [3, 4];

const result1 = arr1.concat(arr2);

console.log("Example 1: Merged numbers:", result1);

// Example 2: Concatenating arrays of strings

const fruits = ['apple', 'banana'];

const berries = ['strawberry', 'blueberry'];

const result2 = fruits.concat(berries);

console.log("\nExample 2: Merged fruits and berries:", result2);

// Example 3: Concatenating with single elements

const base = ['x', 'y'];

const result3 = base.concat('z');

console.log("\nExample 3: Added single element:", result3);

// Example 4: Concatenating nested arrays (does not flatten deeply)

const nested1 = [1, 2];

const nested2 = [[3, 4]];

const result4 = nested1.concat(nested2);

console.log("\nExample 4: Nested array concat:", result4);

// Example 5: Concatenating multiple arrays at once

const a = [10];

const b = [20];

const c = [30];

const d = [40];

const result5 = a.concat(b, c, d);

console.log("\nExample 5: Multiple array concat:", result5);

===================================

Example 1: Merged numbers: [1, 2, 3, 4]

Example 2: Merged fruits and berries: ['apple', 'banana', 'strawberry', 'blueberry']

Example 3: Added single element: ['x', 'y', 'z']

Example 4: Nested array concat: [1, 2, [3, 4]]

Example 5: Multiple array concat: [10, 20, 30, 40]

===================================

// Example 1: Join with default separator (comma)

const letters = ['a', 'b', 'c'];

const result1 = letters.join();

console.log("Example 1: Default separator (comma):", result1);

// Example 2: Join with custom separator ('-')

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

const result2 = numbers.join('-');

console.log("\nExample 2: Hyphen as separator:", result2);

// Example 3: Join with no separator (empty string)

const chars = ['J', 'S'];

const result3 = chars.join('');

console.log("\nExample 3: Join without separator:", result3);

// Example 4: Join array of words into a sentence

const words = ['JavaScript', 'is', 'awesome'];

const result4 = words.join(' ');

console.log("\nExample 4: Words into sentence:", result4);

// Example 5: Join with a symbol (like emoji or special character)

const emojis = ['😀', '🚀', '🌟'];

const result5 = emojis.join(' 💥 ');

console.log("\nExample 5: Join with emoji separator:", result5);

========================================

Example 1: Default separator (comma): a,b,c

Example 2: Hyphen as separator: 1-2-3-4

Example 3: Join without separator: JS

Example 4: Words into sentence: JavaScript is awesome

Example 5: Join with emoji separator: 😀 💥 🚀 💥 🌟

======================================

// Example 1: Find index of a number

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

const result1 = numbers.indexOf(30);

console.log("Example 1: Index of 30:", result1);

// Example 2: Find index of a string

const colors = ['red', 'green', 'blue'];

const result2 = colors.indexOf('green');

console.log("\nExample 2: Index of 'green':", result2);

// Example 3: Element not found

const fruits = ['apple', 'banana', 'cherry'];

const result3 = fruits.indexOf('mango');

console.log("\nExample 3: Index of 'mango' (not found):", result3);

// Example 4: Starting search from specific index

const data = [1, 2, 3, 2, 4];

const result4 = data.indexOf(2, 2);  // Start from index 2

console.log("\nExample 4: Index of 2 starting from index 2:", result4);

// Example 5: Case-sensitive search in strings

const animals = ['Cat', 'Dog', 'cat'];

const result5 = animals.indexOf('cat');

console.log("\nExample 5: Case-sensitive index of 'cat':", result5);

=============================================

Example 1: Index of 30: 2

Example 2: Index of 'green': 1

Example 3: Index of 'mango' (not found): -1

Example 4: Index of 2 starting from index 2: 3

Example 5: Case-sensitive index of 'cat': 2

===============================

// Example 1: Find last index of a repeated number

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

const result1 = numbers.lastIndexOf(2);

console.log("Example 1: Last index of 2:", result1);

// Example 2: Find last index of a string

const names = ['Alice', 'Bob', 'Charlie', 'Bob'];

const result2 = names.lastIndexOf('Bob');

console.log("\nExample 2: Last index of 'Bob':", result2);

// Example 3: Element not found

const fruits = ['apple', 'banana', 'cherry'];

const result3 = fruits.lastIndexOf('mango');

console.log("\nExample 3: Last index of 'mango':", result3);

// Example 4: Start search from a specific index

const values = [1, 2, 3, 2, 4, 2];

const result4 = values.lastIndexOf(2, 3); // Search from index 3 backwards

console.log("\nExample 4: Last index of 2 starting from index 3:", result4);

// Example 5: Case-sensitive string search

const animals = ['Cat', 'Dog', 'cat', 'Dog'];

const result5 = animals.lastIndexOf('Dog');

console.log("\nExample 5: Last index of 'Dog':", result5);

========================================

Example 1: Last index of 2: 5

Example 2: Last index of 'Bob': 3

Example 3: Last index of 'mango': -1

Example 4: Last index of 2 starting from index 3: 3

Example 5: Last index of 'Dog': 3

=================================

// Example 1: Find the first even number

const nums1 = [1, 3, 5, 6, 7];

const result1 = nums1.find(num => num % 2 === 0);

console.log("Example 1: First even number:", result1);

// Example 2: Find the first string longer than 5 characters

const words = ["hi", "hello", "welcome", "JS"];

const result2 = words.find(word => word.length > 5);

console.log("\nExample 2: First long word (>5):", result2);

// Example 3: Find the first negative number

const nums2 = [4, 2, 0, -1, -3];

const result3 = nums2.find(n => n < 0);

console.log("\nExample 3: First negative number:", result3);

// Example 4: Find first object with price > 100

const products = [

  { id: 1, name: "Pen", price: 20 },

  { id: 2, name: "Book", price: 120 },

  { id: 3, name: "Bag", price: 250 },

];

const result4 = products.find(p => p.price > 100);

console.log("\nExample 4: First product with price > 100:", result4);

// Example 5: Find first user with name starting with 'S'

const users = [

  { id: 1, name: "Alice" },

  { id: 2, name: "Steve" },

  { id: 3, name: "Bob" },

];

const result5 = users.find(user => user.name.startsWith("S"));

console.log("\nExample 5: First user name starting with 'S':", result5);

===================================

Example 1: First even number: 6

Example 2: First long word (>5): welcome

Example 3: First negative number: -1

Example 4: First product with price > 100: { id: 2, name: 'Book', price: 120 }

Example 5: First user name starting with 'S': { id: 2, name: 'Steve' }

===================================

// Example 1: Find index of first even number

const numbers1 = [1, 3, 5, 4, 7];

const index1 = numbers1.findIndex(num => num % 2 === 0);

console.log("Example 1: Index of first even number:", index1);

// Example 2: Find index of first string with length > 4

const names = ["Tom", "Sam", "Robert", "Jon"];

const index2 = names.findIndex(name => name.length > 4);

console.log("\nExample 2: Index of first long name (>4):", index2);

// Example 3: Find index of first negative number

const numbers2 = [3, 2, 0, -5, -10];

const index3 = numbers2.findIndex(num => num < 0);

console.log("\nExample 3: Index of first negative number:", index3);

// Example 4: Find index of product with price > 100

const products = [

  { id: 1, name: "Pen", price: 30 },

  { id: 2, name: "Notebook", price: 150 },

  { id: 3, name: "Bag", price: 250 },

];

const index4 = products.findIndex(product => product.price > 100);

console.log("\nExample 4: Index of first product with price > 100:", index4);

// Example 5: Find index of first user whose name starts with "S"

const users = [

  { id: 1, name: "Alice" },

  { id: 2, name: "Steve" },

  { id: 3, name: "Bob" },

];

const index5 = users.findIndex(user => user.name.startsWith("S"));

console.log("\nExample 5: Index of first user starting with 'S':", index5);

=====================================

Example 1: Index of first even number: 3

Example 2: Index of first long name (>4): 2

Example 3: Index of first negative number: 3

Example 4: Index of first product with price > 100: 1

Example 5: Index of first user starting with 'S': 1

==================================

// Example 1: Find last even number

const nums1 = [1, 3, 4, 7, 8];

const result1 = nums1.findLast(n => n % 2 === 0);

console.log("Example 1: Last even number:", result1);

// Example 2: Find last string starting with "J"

const names = ["Tom", "Jerry", "Alice", "Jack", "Jill"];

const result2 = names.findLast(name => name.startsWith("J"));

console.log("\nExample 2: Last name starting with 'J':", result2);

// Example 3: Find last negative number

const nums2 = [10, -1, 20, -2, -3];

const result3 = nums2.findLast(n => n < 0);

console.log("\nExample 3: Last negative number:", result3);

// Example 4: Find last product with price less than 100

const products = [

  { id: 1, name: "Pen", price: 20 },

  { id: 2, name: "Book", price: 150 },

  { id: 3, name: "Notebook", price: 90 },

];

const result4 = products.findLast(p => p.price < 100);

console.log("\nExample 4: Last product with price < 100:", result4);

// Example 5: Find last user with even ID

const users = [

  { id: 1, name: "Amit" },

  { id: 2, name: "Sneha" },

  { id: 3, name: "Ravi" },

  { id: 4, name: "Priya" },

];

const result5 = users.findLast(user => user.id % 2 === 0);

console.log("\nExample 5: Last user with even ID:", result5);

=====================================

Example 1: Last even number: 8

Example 2: Last name starting with 'J': Jill

Example 3: Last negative number: -3

Example 4: Last product with price < 100: { id: 3, name: 'Notebook', price: 90 }

Example 5: Last user with even ID: { id: 4, name: 'Priya' }

=================================

// Example 1: Find index of last even number

const nums1 = [1, 2, 5, 8, 7];

const index1 = nums1.findLastIndex(n => n % 2 === 0);

console.log("Example 1: Index of last even number:", index1);

// Example 2: Find index of last name starting with 'A'

const names = ["Sam", "Alex", "Andy", "Ben"];

const index2 = names.findLastIndex(name => name.startsWith("A"));

console.log("\nExample 2: Index of last name starting with 'A':", index2);

// Example 3: Find index of last negative number

const nums2 = [10, -1, 5, -3, 2];

const index3 = nums2.findLastIndex(n => n < 0);

console.log("\nExample 3: Index of last negative number:", index3);

// Example 4: Find index of last product with price > 100

const products = [

  { id: 1, name: "Pen", price: 20 },

  { id: 2, name: "Book", price: 150 },

  { id: 3, name: "Bag", price: 250 },

];

const index4 = products.findLastIndex(product => product.price > 100);

console.log("\nExample 4: Index of last product with price > 100:", index4);

// Example 5: Find index of last user with even ID

const users = [

  { id: 1, name: "Alice" },

  { id: 3, name: "Bob" },

  { id: 4, name: "Charlie" },

  { id: 2, name: "Diana" },

];

const index5 = users.findLastIndex(user => user.id % 2 === 0);

console.log("\nExample 5: Index of last user with even ID:", index5);

==============================

Example 1: Index of last even number: 3

Example 2: Index of last name starting with 'A': 2

Example 3: Index of last negative number: 3

Example 4: Index of last product with price > 100: 2

Example 5: Index of last user with even ID: 3

==========================

// Example 1: Basic usage with for...of loop

const fruits = ['apple', 'banana', 'cherry'];

console.log("Example 1: Indexes of fruits array:");

for (const key of fruits.keys()) {

  console.log(key); // 0, 1, 2

}

// Example 2: Using keys() with Array.from

const colors = ['red', 'green', 'blue'];

const keysArray = Array.from(colors.keys());

console.log("\nExample 2: keysArray using Array.from:", keysArray); // [0, 1, 2]

// Example 3: Using keys() with sparse array

const sparse = ['a', , 'c'];

console.log("\nExample 3: Sparse array keys:");

for (const key of sparse.keys()) {

  console.log(key); // 0, 1, 2

}

// Example 4: Accessing keys with .next()

const numbers = [10, 20, 30];

const keyIterator = numbers.keys();

console.log("\nExample 4: Using .next():");

console.log(keyIterator.next().value); // 0

console.log(keyIterator.next().value); // 1

console.log(keyIterator.next().value); // 2

// Example 5: Custom loop to show index-value pairs

const animals = ['cat', 'dog', 'lion'];

console.log("\nExample 5: Index and value pairs:");

for (const index of animals.keys()) {

  console.log(`Index ${index} -> ${animals[index]}`);

}

====================================

Example 1: Indexes of fruits array:

0

1

2

Example 2: keysArray using Array.from: [ 0, 1, 2 ]

Example 3: Sparse array keys:

0

1

2

Example 4: Using .next():

0

1

2

Example 5: Index and value pairs:

Index 0 -> cat

Index 1 -> dog

Index 2 -> lion

==============================

// Example 1: Basic usage with for...of

const fruits = ['apple', 'banana', 'cherry'];

console.log("Example 1: Looping values:");

for (const value of fruits.values()) {

  console.log(value); // apple, banana, cherry

}

// Example 2: Using values() with Array.from

const colors = ['red', 'green', 'blue'];

const valuesArray = Array.from(colors.values());

console.log("\nExample 2: Array.from(values):", valuesArray); // [ 'red', 'green', 'blue' ]

// Example 3: Using values() with sparse array

const sparse = ['a', , 'c'];

console.log("\nExample 3: Sparse array values:");

for (const value of sparse.values()) {

  console.log(value); // 'a', undefined, 'c'

}

// Example 4: Accessing values with .next()

const numbers = [10, 20, 30];

const valueIterator = numbers.values();

console.log("\nExample 4: Using .next():");

console.log(valueIterator.next().value); // 10

console.log(valueIterator.next().value); // 20

console.log(valueIterator.next().value); // 30

// Example 5: Values with custom print logic

const animals = ['cat', 'dog', 'lion'];

console.log("\nExample 5: Value with index:");

const iterator = animals.values();

let i = 0;

for (const val of iterator) {

  console.log(`Value at index ${i++}: ${val}`);

}

=========================================

Example 1: Looping values:

apple

banana

cherry

Example 2: Array.from(values): [ 'red', 'green', 'blue' ]

Example 3: Sparse array values:

a

undefined

c

Example 4: Using .next():

10

20

30

Example 5: Value with index:

Value at index 0: cat

Value at index 1: dog

Value at index 2: lion

==================================================

// Example 1: Basic usage with for...of

const fruits = ['apple', 'banana', 'cherry'];

console.log("Example 1: Key-value pairs of fruits:");

for (const [index, value] of fruits.entries()) {

  console.log(index, value); // 0 apple, 1 banana, 2 cherry

}

// Example 2: Using entries() with Array.from

const colors = ['red', 'green', 'blue'];

const entriesArray = Array.from(colors.entries());

console.log("\nExample 2: Array.from(entries):", entriesArray); // [ [0, 'red'], [1, 'green'], [2, 'blue'] ]

// Example 3: Using entries() with sparse array

const sparse = ['a', , 'c'];

console.log("\nExample 3: Sparse array entries:");

for (const [i, val] of sparse.entries()) {

  console.log(`Index ${i}:`, val); // val is undefined at missing index

}

// Example 4: Destructuring entries with custom formatting

const numbers = [100, 200, 300];

console.log("\nExample 4: Destructuring index-value:");

for (const entry of numbers.entries()) {

  const [i, val] = entry;

  console.log(`Index ${i} => Value ${val}`);

}

// Example 5: entries() with while loop and next()

const animals = ['dog', 'cat', 'lion'];

const animalIterator = animals.entries();

console.log("\nExample 5: Using .next():");

let result;

while (!(result = animalIterator.next()).done) {

  const [i, v] = result.value;

  console.log(`Animal at index ${i}: ${v}`);

}

=========================

Example 1: Key-value pairs of fruits:

0 apple

1 banana

2 cherry

Example 2: Array.from(entries): [ [ 0, 'red' ], [ 1, 'green' ], [ 2, 'blue' ] ]

Example 3: Sparse array entries:

Index 0: a

Index 1: undefined

Index 2: c

Example 4: Destructuring index-value:

Index 0 => Value 100

Index 1 => Value 200

Index 2 => Value 300

Example 5: Using .next():

Animal at index 0: dog

Animal at index 1: cat

Animal at index 2: lion

=========================================

// Example 1: Basic array of strings

const fruits = ['apple', 'banana', 'cherry'];

console.log("Example 1:", fruits.toString()); // "apple,banana,cherry"

// Example 2: Array of numbers

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

console.log("Example 2:", numbers.toString()); // "10,20,30,40"

// Example 3: Nested arrays (not deeply flattened)

const nested = [1, [2, 3], 4];

console.log("Example 3:", nested.toString()); // "1,2,3,4" (flattened one level)

// Example 4: Mixed data types

const mixed = ['a', 100, true, null];

console.log("Example 4:", mixed.toString()); // "a,100,true,"

// Example 5: Empty slots in sparse array

const sparse = ['x', , 'z'];

console.log("Example 5:", sparse.toString()); // "x,,z"

===========================================

Example 1: apple,banana,cherry

Example 2: 10,20,30,40

Example 3: 1,2,3,4

Example 4: a,100,true,

Example 5: x,,z

===========================

// Example 1: Basic usage with numbers

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

console.log("Example 1:", numbers.includes(20)); // true

// Example 2: Using fromIndex (start searching from index 2)

console.log("Example 2:", numbers.includes(20, 2)); // false

// Example 3: Case-sensitive check with strings

const fruits = ['Apple', 'Banana', 'Mango'];

console.log("Example 3:", fruits.includes('apple')); // false (case-sensitive)

// Example 4: Checking for boolean value

const mixed = [true, false, 'yes', 1];

console.log("Example 4:", mixed.includes(false)); // true

// Example 5: Check in sparse arrays

const sparse = [1, , 3];

console.log("Example 5:", sparse.includes(undefined)); // true (missing slot treated as undefined)

=======================

Example 1: true

Example 2: false

Example 3: false

Example 4: true

Example 5: true

==================================

// Example 1: Fill the entire array with a value

let arr1 = new Array(5).fill(0);

console.log("Example 1:", arr1); // [0, 0, 0, 0, 0]

// Example 2: Fill part of the array (from index 2 onward)

let arr2 = [1, 2, 3, 4, 5];

arr2.fill(9, 2);

console.log("Example 2:", arr2); // [1, 2, 9, 9, 9]

// Example 3: Fill between specific indexes (2 to 4, 4 is excluded)

let arr3 = [10, 20, 30, 40, 50];

arr3.fill(0, 2, 4);

console.log("Example 3:", arr3); // [10, 20, 0, 0, 50]

// Example 4: Fill with strings

let arr4 = ['a', 'b', 'c', 'd'];

arr4.fill('x', 1, 3);

console.log("Example 4:", arr4); // ['a', 'x', 'x', 'd']

// Example 5: Fill negative indexes (start from -3 to -1)

let arr5 = [1, 2, 3, 4, 5];

arr5.fill(7, -3, -1);

console.log("Example 5:", arr5); // [1, 2, 7, 7, 5]

=======================================

Example 1: [ 0, 0, 0, 0, 0 ]

Example 2: [ 1, 2, 9, 9, 9 ]

Example 3: [ 10, 20, 0, 0, 50 ]

Example 4: [ 'a', 'x', 'x', 'd' ]

Example 5: [ 1, 2, 7, 7, 5 ]

=================================

// Example 1: Flatten one level (default)

const arr1 = [1, 2, [3, 4]];

console.log("Example 1:", arr1.flat()); // [1, 2, 3, 4]

// Example 2: Flatten two levels deep

const arr2 = [1, [2, [3, 4]]];

console.log("Example 2:", arr2.flat(2)); // [1, 2, 3, 4]

// Example 3: Flatten deeply nested array using Infinity

const arr3 = [1, [2, [3, [4, [5]]]]];

console.log("Example 3:", arr3.flat(Infinity)); // [1, 2, 3, 4, 5]

// Example 4: Flatten sparse array (empty slots are removed)

const arr4 = [1, , 3, [4, , 6]];

console.log("Example 4:", arr4.flat()); // [1, 3, 4, 6]

// Example 5: Flatten array with multiple nesting and types

const arr5 = ['a', ['b', ['c', ['d']]], 'e'];

console.log("Example 5:", arr5.flat(3)); // ['a', 'b', 'c', 'd', 'e']

===================================

Example 1: [1, 2, 3, 4]

Example 2: [1, 2, 3, 4]

Example 3: [1, 2, 3, 4, 5]

Example 4: [1, 3, 4, 6]

Example 5: ['a', 'b', 'c', 'd', 'e']

=======================================

// Example 1: Split each word into characters

const words = ['hi', 'bye'];

const result1 = words.flatMap(word => word.split(''));

console.log("Example 1:", result1); // ['h', 'i', 'b', 'y', 'e']

// Example 2: Duplicate each number into a pair

const numbers = [1, 2, 3];

const result2 = numbers.flatMap(n => [n, n]);

console.log("Example 2:", result2); // [1, 1, 2, 2, 3, 3]

// Example 3: Filter and transform at once

const mixed = [1, 2, 3, 4];

const result3 = mixed.flatMap(n => n % 2 === 0 ? [n \* 10] : []);

console.log("Example 3:", result3); // [20, 40]

// Example 4: Flatten array of arrays by 1 level

const nested = [[1], [2, 3], [4]];

const result4 = nested.flatMap(x => x);

console.log("Example 4:", result4); // [1, 2, 3, 4]

// Example 5: Expand items conditionally

const flags = [true, false, true];

const result5 = flags.flatMap(flag => flag ? ['yes'] : []);

console.log("Example 5:", result5); // ['yes', 'yes']

=================================

Example 1: [ 'h', 'i', 'b', 'y', 'e' ]

Example 2: [ 1, 1, 2, 2, 3, 3 ]

Example 3: [ 20, 40 ]

Example 4: [ 1, 2, 3, 4 ]

Example 5: [ 'yes', 'yes' ]

====================================

const sampleArray = ['a', 'b', 'c', 'd', 'e'];

// Example 1: Accessing a positive index

console.log("Example 1:", sampleArray.at(0)); // 'a'

// Example 2: Accessing a middle index

console.log("Example 2:", sampleArray.at(2)); // 'c'

// Example 3: Accessing the last element using negative index

console.log("Example 3:", sampleArray.at(-1)); // 'e'

// Example 4: Accessing the second-to-last element

console.log("Example 4:", sampleArray.at(-2)); // 'd'

// Example 5: Using `at()` with array of numbers

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

console.log("Example 5:", numbers.at(-3)); // 20

==================================

Example 1: a

Example 2: c

Example 3: e

Example 4: d

Example 5: 20

===============================

// Example 1: Getting the length of a basic array

const fruits = ['apple', 'banana', 'mango'];

console.log("Example 1:", fruits.length); // 3

// Example 2: Empty array length

const emptyArr = [];

console.log("Example 2:", emptyArr.length); // 0

// Example 3: Length after pushing new elements

const numbers = [1, 2];

numbers.push(3, 4);

console.log("Example 3:", numbers.length); // 4

// Example 4: Changing length to truncate the array

const letters = ['a', 'b', 'c', 'd'];

letters.length = 2;

console.log("Example 4:", letters); // ['a', 'b']

console.log("Length after truncation:", letters.length); // 2

// Example 5: Creating an empty array with fixed length

const fixedSizeArray = new Array(5);

console.log("Example 5:", fixedSizeArray.length); // 5

=======================================

Example 1: 3

Example 2: 0

Example 3: 4

Example 4: [ 'a', 'b' ]

Length after truncation: 2

Example 5: 5

========================================

// Example 1: Delete an element by index

const fruits = ['apple', 'banana', 'cherry'];

delete fruits[1];

console.log("Example 1:", fruits); // ['apple', <1 empty item>, 'cherry']

console.log("Length:", fruits.length); // 3

// Example 2: Delete the first element

const numbers = [10, 20, 30];

delete numbers[0];

console.log("Example 2:", numbers); // [<1 empty item>, 20, 30]

// Example 3: Delete last element

const colors = ['red', 'green', 'blue'];

delete colors[2];

console.log("Example 3:", colors); // ['red', 'green', <1 empty item>]

// Example 4: Loop and delete even indexes

const letters = ['a', 'b', 'c', 'd', 'e'];

for (let i = 0; i < letters.length; i += 2) {

  delete letters[i];

}

console.log("Example 4:", letters); // [<1 empty item>, 'b', <1 empty item>, 'd', <1 empty item>]

// Example 5: Check array after delete

const items = ['pen', 'pencil', 'eraser'];

delete items[1];

console.log("Example 5:", items.includes('pencil')); // false

console.log("After delete:", items); // ['pen', <1 empty item>, 'eraser']

=======================================

Example 1: [ 'apple', <1 empty item>, 'cherry' ]

Length: 3

Example 2: [ <1 empty item>, 20, 30 ]

Example 3: [ 'red', 'green', <1 empty item> ]

Example 4: [ <1 empty item>, 'b', <1 empty item>, 'd', <1 empty item> ]

Example 5: false

After delete: [ 'pen', <1 empty item>, 'eraser' ]

===========================================

// Example 1: Reverse a simple number array

const arr1 = [1, 2, 3, 4, 5];

console.log("Example 1:", arr1.reverse()); // [5, 4, 3, 2, 1]

// Example 2: Reverse an array of strings

const arr2 = ['a', 'b', 'c'];

console.log("Example 2:", arr2.reverse()); // ['c', 'b', 'a']

// Example 3: Reverse a mixed-type array

const arr3 = [true, 42, 'hello'];

console.log("Example 3:", arr3.reverse()); // ['hello', 42, true]

// Example 4: Reverse a nested array

const arr4 = [[1, 2], [3, 4], [5, 6]];

console.log("Example 4:", arr4.reverse()); // [[5, 6], [3, 4], [1, 2]]

// Example 5: Reverse a copy of the array (without modifying original)

const original = [10, 20, 30];

const reversed = [...original].reverse(); // using spread to copy

console.log("Example 5:", reversed);      // [30, 20, 10]

console.log("Original remains unchanged:", original); // [10, 20, 30]

=================================

Example 1: [5, 4, 3, 2, 1]

Example 2: ['c', 'b', 'a']

Example 3: ['hello', 42, true]

Example 4: [[5, 6], [3, 4], [1, 2]]

Example 5: [30, 20, 10]

Original remains unchanged: [10, 20, 30]

======================================

// Example 1: Basic use with numbers

const arr1 = [1, 2, 3];

const reversed1 = arr1.toReversed();

console.log("Example 1:", reversed1); // [3, 2, 1]

console.log("Original:", arr1);       // [1, 2, 3]

// Example 2: With strings

const arr2 = ['a', 'b', 'c'];

console.log("Example 2:", arr2.toReversed()); // ['c', 'b', 'a']

// Example 3: With mixed types

const arr3 = [true, 'yes', 10];

console.log("Example 3:", arr3.toReversed()); // [10, 'yes', true]

// Example 4: With nested arrays

const arr4 = [[1], [2], [3]];

console.log("Example 4:", arr4.toReversed()); // [[3], [2], [1]]

// Example 5: Compare with reverse()

const arr5 = [100, 200, 300];

const reversed5 = arr5.toReversed();

arr5.reverse(); // Mutates original

console.log("Example 5 - toReversed():", reversed5); // [300, 200, 100]

console.log("Example 5 - reverse():", arr5);         // [300, 200, 100]

====================================

Example 1: [3, 2, 1]

Original: [1, 2, 3]

Example 2: ['c', 'b', 'a']

Example 3: [10, 'yes', true]

Example 4: [[3], [2], [1]]

Example 5 - toReversed(): [300, 200, 100]

Example 5 - reverse(): [300, 200, 100]

=================================

// Example 1: Creating an array with numbers

const numbers = Array.of(1, 2, 3, 4);

console.log("Example 1:", numbers); // [1, 2, 3, 4]

// Example 2: Creating an array with a single number

const single = Array.of(5);

console.log("Example 2:", single); // [5]

// Example 3: Creating an array with mixed types

const mixed = Array.of(1, 'hello', true, null);

console.log("Example 3:", mixed); // [1, 'hello', true, null]

// Example 4: Using Array.of() to wrap existing values (e.g., function return)

function getValue() {

  return 42;

}

const wrapped = Array.of(getValue());

console.log("Example 4:", wrapped); // [42]

// Example 5: Creating an empty array with no arguments

const empty = Array.of();

console.log("Example 5:", empty); // []

==================================

Example 1: [1, 2, 3, 4]

Example 2: [5]

Example 3: [1, 'hello', true, null]

Example 4: [42]

Example 5: []

==================================

// Example 1: Convert a string to an array

const str = "hello";

const chars = Array.from(str);

console.log("Example 1:", chars); // ['h', 'e', 'l', 'l', 'o']

// Example 2: Convert a Set to an array

const mySet = new Set([1, 2, 3]);

const arrFromSet = Array.from(mySet);

console.log("Example 2:", arrFromSet); // [1, 2, 3]

// Example 3: Convert arguments object to array

function example3() {

  const argsArray = Array.from(arguments);

  console.log("Example 3:", argsArray); // ['a', 'b', 'c']

}

example3('a', 'b', 'c');

// Example 4: Create an array of numbers with map function

const doubleNumbers = Array.from([1, 2, 3], x => x \* 2);

console.log("Example 4:", doubleNumbers); // [2, 4, 6]

// Example 5: Generate array with fixed length and initial values

const fiveZeros = Array.from({ length: 5 }, () => 0);

console.log("Example 5:", fiveZeros); // [0, 0, 0, 0, 0]

===================================

Example 1: [ 'h', 'e', 'l', 'l', 'o' ]

Example 2: [ 1, 2, 3 ]

Example 3: [ 'a', 'b', 'c' ]

Example 4: [ 2, 4, 6 ]

Example 5: [ 0, 0, 0, 0, 0 ]

=========================================

// Example 1: Check a basic array

const list = [1, 2, 3];

console.log("Example 1:", Array.isArray(list)); // true

// Example 2: Check a string

const name = "hello";

console.log("Example 2:", Array.isArray(name)); // false

// Example 3: Check an object

const obj = { a: 1, b: 2 };

console.log("Example 3:", Array.isArray(obj)); // false

// Example 4: Check an array created using new Array()

const dynamicArr = new Array(3);

console.log("Example 4:", Array.isArray(dynamicArr)); // true

// Example 5: Check value from JSON

const json = JSON.parse('{"users":["Tom","Jerry"]}');

console.log("Example 5:", Array.isArray(json.users)); // true

console.log("Also check JSON object itself:", Array.isArray(json)); // false

======================================

Example 1: true

Example 2: false

Example 3: false

Example 4: true

Example 5: true

Also check JSON object itself: false

===================================

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

// Example 1: Using Array.prototype.map

const doubled = Array.prototype.map.call(arr, x => x \* 2);

console.log("Example 1 (map):", doubled); // [2, 4, 6, 8, 10]

// Example 2: Using Array.prototype.filter

const even = Array.prototype.filter.call(arr, x => x % 2 === 0);

console.log("Example 2 (filter):", even); // [2, 4]

// Example 3: Using Array.prototype.reduce

const sum = Array.prototype.reduce.call(arr, (acc, val) => acc + val, 0);

console.log("Example 3 (reduce):", sum); // 15

// Example 4: Using Array.prototype.includes

const hasThree = Array.prototype.includes.call(arr, 3);

console.log("Example 4 (includes):", hasThree); // true

// Example 5: Using Array.prototype.forEach

console.log("Example 5 (forEach):");

Array.prototype.forEach.call(arr, x => console.log("  Value:", x));

======================================

Example 1 (map): [2, 4, 6, 8, 10]

Example 2 (filter): [2, 4]

Example 3 (reduce): 15

Example 4 (includes): true

Example 5 (forEach):

  Value: 1

  Value: 2

  Value: 3

  Value: 4

  Value: 5

==============================

// Example 1: Basic usage

const arr1 = [1, 2, 3];

console.log("Example 1:", arr1.valueOf()); // [1, 2, 3]

// Example 2: Comparing array and valueOf() result

const arr2 = ['a', 'b'];

console.log("Example 2:", arr2 === arr2.valueOf()); // true

// Example 3: Use valueOf() in a function

function logValueOf(array) {

  console.log("Example 3:", array.valueOf());

}

logValueOf([10, 20]);

// Example 4: Cloning using valueOf() (shallow, same reference!)

const arr4 = [99, 100];

const ref = arr4.valueOf();

ref[0] = 500;

console.log("Example 4 - modified ref:", ref); // [500, 100]

console.log("Original arr4 also modified:", arr4); // [500, 100]

// Example 5: Custom object with valueOf override (for comparison)

const obj = {

  value: [7, 8, 9],

  valueOf: function () {

    return this.value;

  }

};

console.log("Example 5:", obj.valueOf()); // [7, 8, 9]

==============================

Example 1: [1, 2, 3]

Example 2: true

Example 3: [10, 20]

Example 4 - modified ref: [500, 100]

Original arr4 also modified: [500, 100]

Example 5: [7, 8, 9]

======================================

// Example 1: Basic usage - copy elements starting at index 0 to index 3

const arr1 = [1, 2, 3, 4, 5];

arr1.copyWithin(3, 0, 2);

console.log("Example 1:", arr1); // [1, 2, 3, 1, 2]

// Example 2: Copy from index 1 to end, paste starting at index 0

const arr2 = ['a', 'b', 'c', 'd'];

arr2.copyWithin(0, 1);

console.log("Example 2:", arr2); // ['b', 'c', 'd', 'd']

// Example 3: Copy part of array to index 2

const arr3 = [10, 20, 30, 40, 50];

arr3.copyWithin(2, 0, 2);

console.log("Example 3:", arr3); // [10, 20, 10, 20, 50]

// Example 4: Using negative indices

const arr4 = [1, 2, 3, 4, 5];

arr4.copyWithin(-2, 0, 2);  // copy 0–2 to position -2 (i.e. index 3)

console.log("Example 4:", arr4); // [1, 2, 3, 1, 2]

// Example 5: No end index (copies till the end)

const arr5 = [100, 200, 300, 400, 500];

arr5.copyWithin(1, 3);

console.log("Example 5:", arr5); // [100, 400, 500, 400, 500]

==================================

Example 1: [1, 2, 3, 1, 2]

Example 2: ['b', 'c', 'd', 'd']

Example 3: [10, 20, 10, 20, 50]

Example 4: [1, 2, 3, 1, 2]

Example 5: [100, 400, 500, 400, 500]

============================================

    </script>

</body>

</html>