Lab Report 01

1. Arrays – Basics

Task 1.1: Creating and Accessing Arrays

Code:

```
const fruits = ["Apple", "Banana", "Mango", "Orange", "Grapes"];
console.log(fruits);

// Print first and last item
console.log("First fruit:", fruits[0]);
console.log("Last fruit:", fruits[fruits.length - 1]);

// Add new fruit at the end
fruits.push("Pineapple");

// Remove the second fruit
fruits.splice(1, 1);

// Find the length
console.log("Array length:", fruits.length);
```

Output:

Task 1.2: Looping Through an Array

Code:

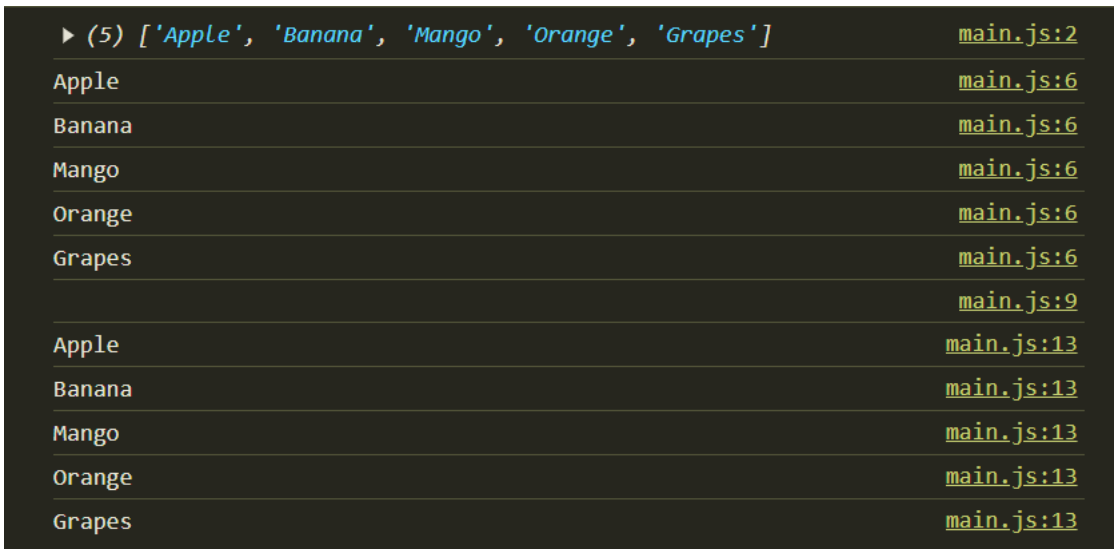
```
const fruits = ["Apple", "Banana", "Mango", "Orange", "Grapes"];
console.log(fruits);

// Using for loop
for (let i = 0; i < fruits.length; i++) {
  console.log(fruits[i]);
}

console.log("");

// Using forEach
fruits.forEach((fruit) => {
  console.log(fruit);
});
```

Output:



difference between using a normal for loop and forEach()

- **For loop:** Provides more control with index access, can use break and continue, suitable when you need the index.
- **forEach():** Simpler and cleaner syntax, automatically iterates through each element, cannot use break or continue.

2. Array Methods

Task 2.1: Transforming Arrays

Code:

```
const numbers = [2, 5, 8, 10, 12];

// Use map() to double each number
const doubled = numbers.map(num => num * 2);
console.log("Doubled:", doubled);

// Use filter() to keep numbers > 6
const filtered = numbers.filter(num => num > 6);
console.log("Filtered:", filtered);

// Use reduce() to find sum
const sum = numbers.reduce((acc, num) => acc + num, 0);
console.log("Sum:", sum);
```

Output:

```
Doubled: ▶ (5) [4, 10, 16, 20, 24]                main.js:5
Filtered: ▶ (3) [8, 10, 12]                       main.js:9
Sum: 37                                           main.js:13
```

Task 2.2: Searching and Sorting

Code:

```
const numbers = [2, 5, 8, 10, 12];

// Check if 8 is in the array
console.log("Includes 8?", numbers.includes(8));

// Find index of 10
console.log("Index of 10:", numbers.indexOf(10));

// Sort in ascending order
const sorted = [...numbers].sort((a, b) => a - b);
console.log("Sorted:", sorted);
```

Output:

```
Includes 8? true                                    main.js:4
Index of 10: 3                                      main.js:7
Sorted:  ▶ (5) [2, 5, 8, 10, 12]                    main.js:11
```

3. Objects – Basics

Task 3.1: Creating and Accessing Objects

Code:

```javascript
const student = {
  name: "Kavindu",
  age: 21,
  faculty: "Computing",
  subjects: ["Web Development", "Database Systems", "Programming"]
};

// Print name and faculty
console.log("Name:", student.name);
console.log("Faculty:", student.faculty);

// Add new property
student.year = 2025;

// Change age
student.age = 22;

// Print all subjects
student.subjects.forEach(subject => {
  console.log("- " + subject);
});
```

Screenshot:

```
Name: Kavindu                                       main.js:9
Faculty: Computing                                  main.js:10
- Web Development                                   main.js:20
- Database Systems                                  main.js:20
- Programming                                       main.js:20
```

4. Nested Objects and Arrays

Task 4.1: Array of Objects

Code:

```
const students = [
  {name: "Kavindu", age: 21, faculty: "Computing"},
  {name: "Nimesha", age: 22, faculty: "Engineering"},
  {name: "Dinuka", age: 23, faculty: "Business"}
];

// Print all names
students.forEach(s => console.log(s.name));

// Filter students older than 21
const olderStudents = students.filter(s => s.age > 21);
console.log(olderStudents);

// Map faculty names
const faculties = students.map(s => s.faculty);
console.log(faculties);
```

Output:

```
Kavindu                                                    main.js:8
Nimesha                                                    main.js:8
Dinuka                                                     main.js:8
▶ (2) [{…}, {…}]                                           main.js:12
▶ (3) ['Computing', 'Engineering', 'Business']             main.js:16
```

Task 4.2: Object Containing an Array of Objects

Code:

```
const classroom = {
  className: "IT2025",
  teacher: "Mr. Perera",
  students: [
    {name: "Kavindu", age: 21},
    {name: "Nimesha", age: 22},
    {name: "Dinuka", age: 23}
  ]
};

// Print teacher name
console.log("Teacher:", classroom.teacher);

// Add new student
classroom.students.push({name: "Sanduni", age: 20});

// Print all student names
classroom.students.forEach(s => console.log(s.name));
```

Output:

```
Teacher: Mr. Perera                                    main.js:12
Kavindu                                                main.js:18
Nimesha                                                main.js:18
Dinuka                                                 main.js:18
Sanduni                                                main.js:18
```

5. Challenge Activity

Code:

```
const products = [
  {name: "Keyboard", price: 2500, qty: 2},
  {name: "Mouse", price: 1500, qty: 3},
  {name: "Monitor", price: 22000, qty: 1}
];

// Calculate total value
const totalValue = products.reduce((total, product) => {
  return total + (product.price * product.qty);
}, 0);

console.log("Total Value: LKR", totalValue);
```

Screenshot:

6. Reflection Questions

1. What is the difference between an array and an object?

- **Array:** An ordered collection of values accessed by numeric indices. Used when order matters and elements are similar.
    - Example: ["Apple", "Banana", "Mango"]
- **Object:** A collection of key value pairs accessed by property names. Used for structured data with different properties.
    - Example: {name: "John", age: 21, faculty: "Computing"}

2. Why are arrays and objects often used together?

- **Arrays of Objects:** Store multiple records with the same structure (e.g., list of students, products)
- **Objects with Array Properties:** Store multiple values for a single property (e.g., a student's subjects)
- This combination creates flexible, powerful data structures that model real-world scenarios effectively
- Examples: JSON data from APIs, database records, application state management

3. What was the most challenging part of this lab?

The most challenging part for me was understanding how reduce() works, especially with the accumulator parameter. It took some time to grasp how the callback function builds up the total value by processing each element sequentially.