Bài tập Javascript cơ bản

Kiểu dữ liệu number trong Javascript (Mandatory)

Bài 1

Viết code thực hiện yêu cầu sau:

Dùng hàm prompt yêu cầu người dùng nhập vào hai số. Tính và hiển thị tổng hai số.

Test cases:

```
// Test case 1Input: 5, 3
Expected output: 8

// Test case 2Input: -1, 7
Expected output: 6
```

Bài 2

Cho 2 số a, b. In ra kết quả của a chia cho b với 2 phần thập phân

Test cases: Cho a = 1; b = 2; In ra "0.50"; Cho a = 1; b = 3; In ra "0.33"; Cho a = 2; b = 1; In ra "2.00";

Bài 3

Viết hàm calculateDiscount(price, percentage) thực hiện:

- Nhận vào giá gốc và phần trăm giảm giá
- Tính và trả về giá sau khi giảm
- Làm tròn kết quả đến 2 chữ số thập phân

Test cases:

```
// Test case 1Input: price = 100, percentage = 20
Expected output: 80.00
```

```
// Test case 2Input: price = 75.50, percentage = 15
Expected output: 64.18
```

Kiểu dữ liệu string trong Javascript(Mandatory)

Bài 1

Viết hàm capitalizeWords(str) thực hiện:

- Nhận vào một chuỗi
- Chuyển chữ cái đầu của mỗi từ thành chữ hoa
- Trả về chuỗi mới

Test cases:

```
// Test case 1Input: "hello world"
Expected output: "Hello World"

// Test case 2Input: "javascript programming"
Expected output: "Javascript Programming"
```

Bài 2

Viết hàm countVowels(str) thực hiện:

- Đếm số nguyên âm trong chuỗi (a, e, i, o, u)
- Không phân biệt chữ hoa/thường
- Trả về số lượng nguyên âm

Test cases:

```
// Test case 1Input: "Hello World"
Expected output: 3
```

```
// Test case 2Input: "JavaScript"
Expected output: 2
```

Bài 3

Viết hàm palindromeCheck(str) thực hiện:

- Kiểm tra chuỗi có phải là palindrome không
- Bỏ qua khoảng trắng và dấu câu
- Trả về true/false

Test cases:

```
// Test case 1Input: "Race a car"
Expected output: false

// Test case 2Input: "A man, a plan, a canal: Panama"
Expected output: true
```

Các phương thức của Object trong Javascript(Mandatory)

Bài 1

Viết hàm deepClone(obj) thực hiện:

- Tạo bản sao sâu của một object (deep clone)
- Xử lý được các object lồng nhau

Test cases:

```
// Test case 1Input: {
  name: "John",
  age: 30,
  address: {
    street: "123 Main St",
    city: "Boston"
```

```
  }
}
```
Expected output: Một object mới với cùng cấu trúc và giá trị, nhưng là instanc
e khác

```
// Test case 2Input: {
  items: [1, 2, {x: 10}],
  info: {
    active: true,
    created: new Date()
  }
}
```
Expected output: Một object mới với cùng cấu trúc và giá trị, nhưng là instanc
e khác

Bài 2

Viết hàm flattenObject(obj) thực hiện:

- Làm phẳng một object có cấu trúc lồng nhau
- Tạo các key mới bằng cách nối các key cũ với dấu "."
- Chỉ làm phẳng các object, giữ nguyên array
- Hint: Sử dụng recursion|stack để làm phẳng một đối tượng lồng nhau

Test cases:

```
// Test case 1Input: {
  name: "John",
  info: {
    age: 30,
    address: {
      city: "New York"
    }
  }
}
```
Expected output: {

```
  "name": "John",
  "info.age": 30,
  "info.address.city": "New York"
}

// Test case 2Input: {
  items: [1, 2, 3],
  details: {
    status: "active",
    meta: {
      created: "2025-01-01",
      updated: "2025-01-02"
    }
  }
}
Expected output: {
  "items": [1, 2, 3],
  "details.status": "active",
  "details.meta.created": "2025-01-01",
  "details.meta.updated": "2025-01-02"
}
```

Các phương thức của Array trong Javascript(Mandatory)

Bài 1

Viết hàm filterDuplicates(arr) thực hiện:

- Loại bỏ các phần tử trùng lặp trong mảng
- Giữ nguyên thứ tự các phần tử
- Trả về mảng mới

Test cases:

```
// Test case 1Input: [1, 2, 3, 3, 4, 4, 5]
Expected output: [1, 2, 3, 4, 5]

// Test case 2Input: ["a", "b", "a", "c", "b"]
Expected output: ["a", "b", "c"]
```

Bài 2

Viết hàm groupByProperty(arr, prop) thực hiện:

- Nhận vào mảng objects và tên thuộc tính
- Nhóm các object theo giá trị của thuộc tính
- Trả về object với key là giá trị thuộc tính
- Hint: Use reduce, push, forEach

Test cases:

```
// Test case 1Input:
arr = [{type: "fruit", name: "apple"}, {type: "vegetable", name: "carrot"}, {type: "fruit", name: "banana"}]
prop = "type"
Expected output: {
  fruit: [{type: "fruit", name: "apple"}, {type: "fruit", name: "banana"}],
  vegetable: [{type: "vegetable", name: "carrot"}]
}

// Test case 2Input:
arr = [{age: 20, name: "John"}, {age: 20, name: "Jane"}, {age: 25, name: "Tom"}]
prop = "age"
Expected output: {
  20: [{age: 20, name: "John"}, {age: 20, name: "Jane"}],
  25: [{age: 25, name: "Tom"}]
}
```

Bài 3

Viết hàm findMostFrequent(arr) thực hiện:

- Tìm phần tử xuất hiện nhiều nhất trong mảng
- Nếu có nhiều phần tử cùng tần suất, trả về phần tử đầu tiên
- Trả về object chứa phần tử và số lần xuất hiện

Test cases:

```
// Test case 1Input: [1, 2, 3, 2, 4, 2, 5]
Expected output: { element: 2, count: 3 }

// Test case 2Input: ["a", "b", "a", "c", "b", "a"]
Expected output: { element: "a", count: 3 }
```

Ứng dụng reduce trong mảng(Mandatory)

Bài 1

Viết hàm calculateCart(items) thực hiện:

- Tính tổng giá trị giỏ hàng
- Áp dụng giảm giá nếu có
- Sử dụng reduce để tính toán

Test cases:

```
// Test case 1Input: [{price: 100, discount: 20}, {price: 50, discount: 0}]
Expected output: 130

// Test case 2Input: [{price: 200, discount: 10}, {price: 300, discount: 15}, {pric
e: 100, discount: 5}]
Expected output: 527.5
```

Bài 2

Viết hàm groupByCategories(transactions) thực hiện:

- Nhóm các giao dịch theo category
- Tính tổng amount cho mỗi category
- Sử dụng reduce

Test cases:

```
// Test case 1Input: [
  {category: "food", amount: 50},
  {category: "transport", amount: 30},
  {category: "food", amount: 20}
]
Expected output: {
  food: 70,
  transport: 30
}

// Test case 2Input: [
  {category: "bills", amount: 100},
  {category: "food", amount: 50},
  {category: "bills", amount: 50}
]
Expected output: {
  bills: 150,
  food: 50
}
```

Bài 3

Viết hàm calculateStats(numbers) thực hiện:

- Tính min, max, average của mảng số
- Sử dụng một lần reduce duy nhất
- Trả về object chứa các giá trị thống kê

Test cases:

```
// Test case 1Input: [1, 2, 3, 4, 5]
Expected output: {
  min: 1,
  max: 5,
  average: 3
}

// Test case 2Input: [10, 20, 30, 40]
Expected output: {
  min: 10,
  max: 40,
  average: 25
}
```

Bài tập tổng hợp (Optional)

Bài 1: Quản lý thư viện sách

Viết class Library thực hiện các chức năng:

- Thêm/xóa/sửa sách (dùng array methods)
- Tìm kiếm sách theo tên/tác giả (dùng string methods)
- Thống kê số sách theo thể loại (dùng reduce)
- Sắp xếp sách theo giá/năm xuất bản
- Lưu trữ dữ liệu vào localStorage

Test cases:

```
// Test case 1Input:
const lib = new Library();
lib.addBook({
  id: 1,
  title: "JavaScript Basic",
```

```
  author: "John Doe",
  price: 29.99,
  year: 2025,
  category: "Programming"
});
lib.addBook({
  id: 2,
  title: "Python for Beginners",
  author: "Jane Smith",
  price: 24.99,
  year: 2024,
  category: "Programming"
});
lib.searchBooks("JavaScript");

Expected output: [{
  id: 1,
  title: "JavaScript Basic",
  author: "John Doe",
  price: 29.99,
  year: 2025,
  category: "Programming"
}]

// Test case 2Input:
lib.getCategoryStats();
Expected output: {
  "Programming": 2
}
```

Bài 2: Shopping Cart Manager

Viết class ShoppingCart thực hiện:

- Thêm/xóa/cập nhật số lượng sản phẩm
- Tính tổng tiền có tính đến giảm giá (dùng reduce)

- Lọc sản phẩm theo khoảng giá
- Tìm kiếm sản phẩm
- Lưu giỏ hàng vào localStorage

Test cases:

```
// Test case 1Input:
const cart = new ShoppingCart();
cart.addItem({
  id: 1,
  name: "Laptop",
  price: 999.99,
  quantity: 1,
  discount: 10
});
cart.addItem({
  id: 2,
  name: "Mouse",
  price: 29.99,
  quantity: 2,
  discount: 0
});
cart.getTotal();
Expected output: 959.97// (999.99 - 10%) + (29.99 * 2)// Test case 2Input:
cart.filterByPriceRange(20, 100);
Expected output: [{
  id: 2,
  name: "Mouse",
  price: 29.99,
  quantity: 2,
  discount: 0
}]
```

Bài 3: Task Manager

Viết class TaskManager thực hiện:

- Thêm/xóa/cập nhật task
- Phân loại task theo trạng thái
- Tìm kiếm task theo tên/mô tả
- Thống kê số task theo trạng thái
- Sắp xếp task theo deadline/priority
- Lưu trữ vào localStorage

Test cases:

```
// Test case 1Input:
const tm = new TaskManager();
tm.addTask({
  id: 1,
  title: "Learn JavaScript",
  description: "Study basic concepts",
  status: "in-progress",
  priority: "high",
  deadline: "2025-06-01"
});
tm.addTask({
  id: 2,
  title: "Build Project",
  description: "Create portfolio website",
  status: "todo",
  priority: "medium",
  deadline: "2025-07-01"
});
tm.getTasksByStatus("in-progress");
Expected output: [{
  id: 1,
  title: "Learn JavaScript",
  description: "Study basic concepts",
  status: "in-progress",
```

```
  priority: "high",
  deadline: "2025-06-01"
}]

// Test case 2Input:
tm.getStatusStats();
Expected output: {
  "in-progress": 1,
  "todo": 1
}
```