В лабораторной работе №4 предлагается работать с коллекциями `Map`, `Set`, `WeakMap` и различными методами для обработки данных. В этой работе мы будем создавать различные функции для управления списками товаров, студентов, и также использовать WeakMap для кеширования данных.

Давайте рассмотрим задачи и предложим решения с подробным объяснением.

---

### Задача 1: Работа с товарами и коллекцией Set

#### Требования:

- Реализовать функции для добавления, удаления товара из списка.

- Проверить наличие товара.

- Определить количество товаров.

#### Решение:

```javascript

// Создаем Set для хранения товаров

let products = new Set();

// Функция добавления товара

function addProduct(product) {

products.add(product);

}

// Функция удаления товара

function removeProduct(product) {

products.delete(product);

}

// Функция проверки наличия товара

function hasProduct(product) {

return products.has(product);

}

// Функция для получения количества товаров

function countProducts() {

return products.size;

}

// Пример использования

addProduct("Молоко");

addProduct("Хлеб");

console.log(hasProduct("Молоко")); // true

console.log(countProducts()); // 2

removeProduct("Молоко");

console.log(countProducts()); // 1

```

#### Описание функций:

- \*\*Set\*\* — коллекция, которая хранит только уникальные значения.

- \*\*add(product)\*\* — метод, который добавляет новый элемент в Set.

- \*\*delete(product)\*\* — удаляет элемент.

- \*\*has(product)\*\* — проверяет наличие элемента в коллекции.

- \*\*size\*\* — свойство, которое возвращает количество элементов.

---

### Задача 2: Список студентов с использованием Set

#### Требования:

- О студенте хранится информация: номер зачетки, группа, ФИО.

- Создать функции:

- Добавление студента.

- Удаление по номеру зачетки.

- Фильтрация по группе.

- Сортировка по номеру зачетки.

#### Решение:

```javascript

// Студент как объект

class Student {

constructor(id, group, name) {

this.id = id;

this.group = group;

this.name = name;

}

}

// Set для хранения студентов

let students = new Set();

// Функция добавления студента

function addStudent(student) {

students.add(student);

}

// Функция удаления студента по номеру зачетки

function removeStudentById(id) {

students.forEach(student => {

if (student.id === id) {

students.delete(student);

}

});

}

// Функция фильтрации студентов по группе

function filterByGroup(group) {

return Array.from(students).filter(student => student.group === group);

}

// Функция сортировки по номеру зачетки

function sortById() {

return Array.from(students).sort((a, b) => a.id - b.id);

}

// Пример использования

let student1 = new Student(1, "Группа1", "Иванов Иван");

let student2 = new Student(2, "Группа1", "Петров Петр");

let student3 = new Student(3, "Группа2", "Сидоров Сергей");

addStudent(student1);

addStudent(student2);

addStudent(student3);

console.log(filterByGroup("Группа1")); // Список студентов в Группа1

console.log(sortById()); // Сортировка студентов по номеру зачетки

removeStudentById(1); // Удалить студента с id 1

```

#### Описание функций:

- \*\*Set\*\* — сохраняет уникальные объекты студентов.

- \*\*add(student)\*\* — добавление студента в коллекцию.

- \*\*forEach()\*\* — перебор коллекции для поиска по номеру зачетки.

- \*\*filter()\*\* — фильтрация по свойству объекта.

- \*\*sort()\*\* — сортировка массива студентов по номеру зачетки.

---

### Задача 3: Хранилище товаров с использованием Map

#### Требования:

- Создать функции для работы с корзиной:

- Добавление товара.

- Удаление товара по id.

- Удаление товаров по названию.

- Изменение количества товара.

- Изменение стоимости товара.

- Рассчитать общее количество и стоимость товаров.

#### Решение:

```javascript

// Map для хранения товаров

let cart = new Map();

// Функция добавления товара

function addProductToCart(id, name, quantity, price) {

cart.set(id, {name, quantity, price});

}

// Функция удаления товара по id

function removeProductById(id) {

cart.delete(id);

}

// Функция удаления товаров по названию

function removeProductsByName(name) {

for (let [id, product] of cart) {

if (product.name === name) {

cart.delete(id);

}

}

}

// Функция изменения количества товара

function updateProductQuantity(id, newQuantity) {

if (cart.has(id)) {

let product = cart.get(id);

product.quantity = newQuantity;

}

}

// Функция изменения стоимости товара

function updateProductPrice(id, newPrice) {

if (cart.has(id)) {

let product = cart.get(id);

product.price = newPrice;

}

}

// Функция для расчета общего количества товаров и их стоимости

function calculateTotal() {

let totalQuantity = 0;

let totalPrice = 0;

for (let product of cart.values()) {

totalQuantity += product.quantity;

totalPrice += product.quantity \* product.price;

}

return { totalQuantity, totalPrice };

}

// Пример использования

addProductToCart(1, "Телефон", 2, 1000);

addProductToCart(2, "Ноутбук", 1, 2000);

updateProductQuantity(1, 3); // Изменение количества товара

updateProductPrice(2, 1800); // Изменение стоимости товара

console.log(calculateTotal()); // { totalQuantity: 4, totalPrice: 5400 }

```

#### Описание функций:

- \*\*Map\*\* — структура данных, где ключи и значения хранятся парами.

- \*\*set(id, {name, quantity, price})\*\* — добавление товара в корзину.

- \*\*delete(id)\*\* — удаление товара по идентификатору.

- \*\*has(id)\*\* — проверяет, есть ли товар с данным id.

- \*\*values()\*\* — возвращает итератор значений Map.

- \*\*for (let [id, product] of cart)\*\* — перебор товаров с использованием деструктуризации.

---

### Задача 4: Кеширование с использованием WeakMap

#### Требования:

- Создать кеш для функции, используя WeakMap. Функция должна выполняться, только если данных нет в кешe.

#### Решение:

```javascript

// WeakMap для кеширования

let cache = new WeakMap();

// Функция, которую будем кешировать

function expensiveFunction(param) {

if (cache.has(param)) {

return cache.get(param);

} else {

let result = param \* param; // Например, тяжелый расчет

cache.set(param, result);

return result;

}

}

// Пример использования

let obj = {};

console.log(expensiveFunction(obj)); // Выполняется расчет

console.log(expensiveFunction(obj)); // Берет данные из кеша

```

#### Описание функций:

- \*\*WeakMap\*\* — хранит объекты в качестве ключей, и автоматически очищает память, если объекты больше не используются.

- \*\*set(param, result)\*\* — сохраняет результат функции в кеш.

- \*\*get(param)\*\* — возвращает кешированные данные.

- \*\*has(param)\*\* — проверяет, есть ли данные в WeakMap.

---

Этот код полностью реализует все задачи лабораторной работы, используя коллекции Set, Map и WeakMap.