**1. В чем преимущества использования Map и Set по сравнению с обычными массивами и объектами?**

**Map**:

Сохраняет порядок вставки элементов

Ключи могут быть любого типа (включая объекты), а не только строки или символы.

Есть встроенные методы

Лучшая производительность при частых добавлениях и удалениях.

**Set**:

Хранит только уникальные значения

Быстрый поиск элемента

Удобен для операций с множествами

**2. Каковы основные методы, доступные для объектов Map и Set, и как их использовать для обработки данных?**

**Map**:

const map = new Map();

map.set('key', 'value');

map.get('key');

map.has('key');

map.delete('key');

map.size;

map.clear();

**Set**:

const set = new Set();

set.add('value');

set.has('value');

set.delete('value');

set.size;

set.clear();

**3. Как можно реализовать поиск и фильтрацию данных с помощью методов коллекций?**

**Map**:

const map = new Map([[1, 'a'], [2, 'b']]);

const filtered = new Map([...map].filter(([key, value]) => key > 1));

**Set**:

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

const hasTwo = set.has(2);

const filtered = new Set([...set].filter(x => x > 1));

**4. Какие сценарии использования Set вы можете привести, когда необходимо хранить уникальные значения?**

Удаление дубликатов из массива:

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

Проверка уникальности значений:

const tags = new Set();

tags.add('js');

tags.add('js'); Не добавится

Операции с множествами (объединение, пересечение):

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

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

const union = new Set([...setA, ...setB]);

const intersection = new Set([...setA].filter(x => setB.has(x)));

#### ****1.**** Каковы основные принципы объектно-ориентированного программирования (ООП)?

**Инкапсуляция** — скрытие внутренней реализации и предоставление интерфейса.

**Наследование** — возможность создавать дочерние классы на основе родительских.

**Полиморфизм** — использование одного интерфейса для разных типов данных.

**Абстракция** — упрощение сложных систем путем выделения ключевых свойств.

**2. В чем разница между классом и дочерним классом в JavaScript?**

**Класс** — это шаблон для создания объектов.

**Дочерний класс** (подкласс) наследует свойства и методы родительского класса и может расширять или переопределять их.

**3. Как реализовать несколько конструкторов в классе и какие подходы для этого существуют?**

В JavaScript нельзя иметь несколько конструкторов напрямую, но можно использовать: **Параметры по умолчанию, Фабричные методы**

**4. Какие методы (например, для добавления свойств или отображения информации) вы планируете реализовать в своих классах?**

Примеры методов:

**Добавление свойств**:

class User {

setEmail(email) { this.email = email; }

}

**Отображение информации**:

class Product {

displayInfo() { return `${this.name} - $${this.price}`; }

}

**Валидация данных**:

class Account {

validatePassword(pass) { return pass.length >= 8; }

}