# Лаба 6–7: Ответы на вопросы

## В чем преимущества использования Map и Set по сравнению с обычными массивами и объектами?

• Быстрый доступ: операции поиска в Map/Set выполняются за O(1), тогда как в массивах в среднем O(n).

• Чёткая семантика: Map предназначен для хранения пар «ключ‑значение», а Set — для уникальных значений.

• Типы ключей: в Map ключом может быть любой объект (в объекте — только строка или символ).

• Поддержка размера: свойство size отражает реальное количество элементов без перебора.

• Отсутствие «лишних» свойств: нет наследуемых свойств прототипа, что упрощает итерацию.

• Сохранение порядка вставки: и Map, и Set обходятся в том порядке, в котором элементы были добавлены.

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

Map

• set(key, value) — добавить/обновить пару.

• get(key) — вернуть значение.

• has(key) — проверить наличие.

• delete(key) / clear() — удалить один или все элементы.

• keys() / values() / entries() — итерируемые коллекции для обхода.

• forEach(cb) — обработка каждой записи.

Set

• add(value) — добавить элемент.

• has(value) — проверить наличие.

• delete(value) / clear() — удалить.

• values() / keys() / entries() — вернуть итераторы (ключ = значение).

• forEach(cb) — перебор элементов.

Все методы возвращают сам объект, что позволяет строить цепочки вызовов.

## Как можно реализовать поиск и фильтрацию данных с помощью методов коллекций?

1. Поиск

if (phoneBook.has('Ivan')) { ... } // Set или Map

const price = goods.get('id‑123'); // Map

```

2. Фильтрация (превращаем во временный массив, применяем Array.filter, собираем обратно):

const expensive = new Map([...goods]

.filter(([id, g]) => g.price > 1000));

const even = new Set([...numbers].filter(n => n % 2 === 0));

3. Комбинированный запрос

for (const [k, v] of users.entries()) {

if (k.startsWith('admin\_') && v.active) ...

}

## Какие сценарии использования Set вы можете привести, когда необходимо хранить уникальные значения?

• Хранение посещённых узлов при обходе графа/лабиринта.

• Кеширование уникальных запросов API.

• Дедупликация входных данных (массив → Set → обратно массив без дублей).

• Отслеживание подключённых клиентов по socket‑id.

• Список тегов/хэштегов, которые не должны повторяться.

• Устранение повторных событий (например, debounce кликов).

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

1. Инкапсуляция — объединяем данные и методы, скрывая внутренние детали.

2. Абстракция — выделяем только существенные характеристики.

3. Наследование — создаём новые классы на основе существующих.

4. Полиморфизм — единый интерфейс, разные реализации (методы переопределяются).

## В чем разница между классом и дочерним классом в JavaScript?

Класс (base/parent) — объявляется через `class` и описывает общие свойства/методы.

Дочерний класс (sub/child) — объявляется с помощью `extends`, наследует всё от родителя, может:

• добавлять новые свойства/методы;

• переопределять родительские (override);

• вызывать `super()` в конструкторе и `super.method()` внутри методов.

## Как реализовать несколько конструкторов в классе и какие подходы для этого существуют?

JavaScript не поддерживает перегрузку конструкторов напрямую. Возможные приёмы:

1. Параметры по умолчанию и проверка типа

constructor(a = 0, b = 0) { ... }

2. Объект‑параметр (options)

constructor({ x = 0, y = 0 } = {}) { ... }

3. Статические фабричные методы

static fromPolar(r, fi) { return new Point(r\*Math.cos(fi), r\*Math.sin(fi)); }

4. Функция‑обёртка — возвращает экземпляр нужного класса в зависимости от контекста.

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

• getters/setters — контролируемый доступ к приватным свойствам.

• addItem / removeItem — изменение коллекций внутри объекта.

• toString / toJSON — сериализация и удобный вывод.

• render() / display() — интеграция с UI.

• static validate() — проверка входных данных перед созданием экземпляра.

• Метод‑цепочка `update(props)` возвращает `this` для fluent‑API.