**Объекты**

**Объект — это** комплексный тип данных в JavaScript, который может содержать как данные, так и функции (методы) для работы с этими данными. Объекты могут быть использованы для представления различных сущностей в вашей программе. Они создаются с использованием фигурных скобок {}.

**const person = {**

**firstName: "John",**

**lastName: "Doe",**

**age: 30**

**};**

В приведенном выше примере **person** - это объект, а **firstName**, **lastName** и **age** - его свойства.

**Создание объекта:**

Объекты можно создать двумя основными способами: с использованием **литерала объекта (рекомендованный)** или **конструктора объекта**.

1. **литерал объекта:**

**const car = {**

**make: "Toyota",**

**model: "Camry",**

**year: 2022**

**};**

1. **конструктор объекта**

**const book = new Object();**

**book.title = "JavaScript Basics";**

**book.author = "John Smith";**

**Доступ к свойствам**

Чтобы получить доступ к свойствам объекта, вы можете использовать точечную нотацию или квадратные скобки.

**const person = {**

**firstName: "John",**

**lastName: "Doe",**

**age: 30**

**};**

**const cityName = person.firstName; //** Используем точечную нотацию

**const city = person["lastName"]; //** Используем квадратные скобки

**Изменение и добавление свойств**

Свойства объекта можно изменять или добавлять после его создания.

**const person = {**

**firstName: "John",**

**lastName: "Doe",**

**age: 30**

**};**

**person.age = 31; //** Изменение свойства

**person.city = "New York"; //** Добавление нового свойства

**Удаление свойств**

Свойства объекта можно удалить с помощью оператора delete.

**const person = {**

**firstName: "John",**

**lastName: "Doe",**

**age: 30**

**};**

**delete person.age; //** Удалить свойство "age"

**Перебор свойств объекта:**

Для перебора свойств объекта можно использовать цикл for...in.

**for (let key in person) {**

**console.log(key + ": " + person[key]);**

**}**

**Деструктуризация объекта (Object Destructuring)**

**Деструктуризация объекта -** способ извлечения значений из объекта и присвоения их переменным с использованием синтаксиса, похожего на объявление объекта. Деструктуризация объекта позволяет упростить код, делая его более читаемым и удобным.

Вот базовый синтаксис деструктуризации объекта:

**const { свойство1, свойство2, ... } = объект;**

**Где:**

**{}** - используются фигурные скобки для определения, какие свойства объекта мы хотим извлечь.

**свойство1, свойство2, ...** - это переменные, в которые будут сохранены значения соответствующих свойств объекта.

объект - объект, из которого вы хотите извлечь значения

**const person = { name: "John", age: 30, city: "New York" };**

**const { name, age } = person;**

**console.log(name); //** "John**"**

**console.log(age);** // 30

**Задача 1:** Создайте объект person с полями name, age и city. Затем выведите информацию о этом человеке в консоль.

**Задача 2:** Создайте объект car с полями make, model и year. Измените год выпуска автомобиля и выведите информацию о нем в консоль.

**Задача 3:** Создайте пустой объект student и добавьте в него поля name, age и subjects (массив предметов, которые он изучает).

**Задача 4:** Дан объект book с полями title, author и year. Выведите в консоль каждое из этих полей, используя цикл for...in.

**Задача 5:** Создайте функцию countProperties(obj), которая принимает объект и возвращает количество его свойств.

**Задача 7:** Создайте объект person с полями name, age и city. Сделайте функцию printPersonInfo(person), которая принимает объект и выводит информацию о нем в формате "Имя: [name], Возраст: [age], Город: [city]".

**Задача 8:** Создайте объект product с полями name, price и quantity. Напишите функцию calculateTotalPrice(product), которая принимает объект product и возвращает общую стоимость товара (цена \* количество).

**Конструктор функций**

В JavaScript конструкторы функций используются для создания объектов с одинаковой структурой и методами. Они предоставляют шаблон для создания новых объектов, что упрощает управление и обслуживание кода. Вот несколько основных целей использования конструкторов функций в JavaScript:

1. **Создание экземпляров объектов** - Конструкторы позволяют создавать множество объектов с одинаковой структурой. Например, если у вас есть множество пользователей в веб-приложении, вы можете использовать конструктор, чтобы создавать объекты, представляющие каждого пользователя.
2. **Абстракция данных**: Конструкторы помогают абстрагировать (скрывать) детали реализации объекта. Вы можете определить свойства и методы объекта, а затем создавать новые объекты на основе этой абстракции.
3. **Расширение и наследование:** Вы можете создавать конструкторы, которые служат как базовые классы, и затем наследовать их свойства и методы, создавая дочерние классы. Это полезно для создания иерархии объектов.
4. **Повторное использование кода:** Конструкторы позволяют определить шаблон для объектов, что упрощает повторное использование кода. Вы можете создавать множество объектов на основе одного и того же конструктора.
5. **Упрощение управления состоянием**: Конструкторы могут содержать свойства, которые представляют состояние объекта, и методы для изменения этого состояния. Это упрощает управление данными и их изменением внутри объекта.

**Создание конструктора функций и объектов на основе конструктора**

**function User(name, email) {**

**this.name = name;**

**this.email = email;**

**}**

**// Создание экземпляров объектов**

**const user1 = new User("Анна", "anna@example.com");**

**const user2 = new User("Иван", "ivan@example.com");**