Домашнее задание к лекции 3.1 «Создание конструктора и прототипа»

Перед началом работы

1.Активируйте строгий режим соответствия.

2.Добавьте в редактор следующий блок кода:

```
function rand(min, max) {
  return Math.ceil((max - min + 1) * Math.random()) + min - 1;
}

function generateId() {
  return Array(4).fill(1).map(value => rand(1000, 9999)).join('-');
}

const pointsInfo = [
  { title: 'Темная сторона Луны', coords: [500, 200, 97] },
  { title: 'Седьмое кольцо Юпитера', coords: [934, -491, 712] },
  { title: 'Саратов', coords: [30, 91, 77] }
];
```

Задача № 1. Пункты телепортации заказов.

Мы хотим запустить на нашем сайте сервис, который позволяет пользователю найти ближайший к нему пункт телепортации заказов.

Для этого нужно создать конструктор и прототип для пункта телепортации, который упростит их поиск. Экземпляр должен иметь название и координаты. А также метод, позволяющий вычислить расстояние до него.

Описание конструктора и экземпляра

Конструктор

Принимает четыре аргумента:

1.`title` — название пункта, строка;
2.x, y и z — координаты, числа.

Экземпляр

Имеет всего один метод:

`getDistance` — принимает три координаты точки x, y и z и возвращает расстояние от этой точки до расположения пункта телепортации.

Пример использования

```
const point = new OrdersTeleportationPoint('Темная сторона Луны', 500, 200,
97);
let distance = point.getDistance(100, -100, 33);
console.log(`Расстояние до пункта «${point.title}» составит $
{distance.toFixed(0)} единиц`);
```

Если все реализовано верно, вы получите такой вывод:

```
Расстояние до пункта «Темная сторона Луны» составит 504 единиц
```

Процесс реализации

1. Создайте конструктор `OrdersTeleportationPoint`.
2. Сохраните в экземпляр название и координаты.
3. Реализуйте метод `getDistance`.

Проверьте работу вашего кода на примере использования. Также протестируйте его, используя свои примеры.

Задача № 2. Поиск ближайшего пункта телепортации.

Реализуйте конструктор и прототип объекта, выполняющего поиск ближайшего пункта телепортации.

Описание конструктора и экземпляра

Конструктор

Принимает список пунктов телепортации.

1. `points` — список пунктов телепортации, массив объектов `OrdersTeleportationPoint`.

Если будет передан не массив, то конструктор должен бросить исключение.

Если в массиве будут объекты, не являющиеся экземпляром `OrdersTeleportationPoint`, то их нужно просто проигнорировать.

Экземпляр

Имеет всего один метод:

getClosest — принимает три координаты точки x, y и z и возвращает ближайший к этой точке пункт телепортации.

Пример использования функции

```
const points = pointsInfo.map(point => new
OrdersTeleportationPoint(point.title,...point.coords));
const locator = new OrdersTeleportationPointLocator(points);

const closestPoint = locator.getClosest(333, 294, 77);
console.log(`Ближайший пункт телепортации заказов «${closestPoint.title}»`);
```

Если все реализовано верно, вы получите такой вывод:

```
Ближайший пункт телепортации заказов «Темная сторона Луны»
```

Процесс реализации

1. Создайте конструктор `OrdersTeleportationPointLocator`.
2. Реализуйте метод `getClosest`.

Проверьте работу, запустив пример использования. А также протестируйте, используя собственные примеры.

Задача № 3. Карты лояльности.

Со следующего месяца мы решили внедрить карты лояльности. И нужно подготовить код нашего интернет-магазина к удобной обработке этих карт с использованием правильного прототипно-ориентированного подхода.

Нужно создать конструктор и прототип карты лояльности, который будет отвечать следующим требованиям:

- При создании карты лояльности она получает уникальный номер, который доступен в свойстве id. Его нельзя изменить. Для генерации номера используйте функцию generateId
- Карта создается только при первой покупке, и нужно передать имя заказчика и сумму заказа для её создания.
- У созданной карты должны быть свойства с общей суммой покупок и текущей скидкой в процентах.
- Сумма скидки зависит от общей суммы покупок (включительно):
  - **до 3000 Q** — 0 %;
  - **до 5000 Q** — 3 %;
  - **до 10000 Q** — 5 %;

- **больше 10000 Q** — 7 %.

- У карты должны быть метод получения суммы с учетом скидки по сумме заказа, метод увеличения баланса карты и метод, показывающий информацию по карте.
  Информацию о карте выводить в таком формате:

```
Карта 3929-1248-1760-6564:
  Владелец: Иванов Иван
  Баланс: 13300 Q
  Текущая скидка: 7 %
  Заказы:
    #1 на сумму 6300 Q
    #2 на сумму 7000 Q
```

Описание конструктора и экземпляра

Конструктор

Конструктор должен зафиксировать в экземпляре имя владельца карты и сумму заказа. А также сгенерировать и сохранить идентификатор карты с помощью функции generateId. Должен принимать следующие аргументы:

1. `name` — имя владельца, строка;
2. `sum` — сумма заказа, число.

Экземпляр

Созданные объекты карт лояльности должны иметь следующие свойства и методы:

`owner` — имя владельца карты, строка;
`id` — идентификатор карты, строка, **только чтение**;
`balance` — общая сумма покупок, число, **только чтение**;
`discount` — текущая скидка в процентах, число, **только чтение**;
`getFinalSum` — метод принимает сумму заказа (число) и возвращает сумму с учетом скидки (число);
`append` — метод принимает сумму заказа (число) и увеличивает баланс карты;
`show` — метод не принимает аргументов и выводит в консоль информацию по карте.

Пример использования конструктора и прототипа

```
const card = new LoyaltyCard('Иванов Иван', 6300);

let newOrderSum = 7000;
let finalSum = card.getFinalSum(newOrderSum);
console.log(`Итоговая сумма для заказа на ${newOrderSum} Q по карте
  составит ${finalSum} Q. Скидка ${card.discount} %.`);
```

```
card.append(newOrderSum);
console.log(`Баланс карты после покупки ${card.balance} Q.`);
card.show();
```

Если конструктор и прототип реализованы правильно, то этот пример даст такой вывод:

```
Итоговая сумма для заказа на 7000 Q по карте составит 6650 Q. Скидка 5 %.
Баланс карты после покупки 13300 Q.
Карта 4311-1715-5080-6340:
  Владелец: Иванов Иван
  Баланс: 13300 Q
  Текущая скидка: 7 %
  Заказы:
    #1 на сумму 6300 Q
    #2 на сумму 7000 Q
```

Процесс реализации

1. Создайте конструктор, сгенерируйте идентификатор карты и сохраните имя владельца и сумму заказа.
2. Не забудьте сделать свойство `id` экземпляра только для чтения.
3. Добавьте свойства `balance` и `discount`. Они тоже только для чтения.
4. Реализуйте метод `getFinalSum`.
5. Реализуйте метод `append`.
6. Реализуйте метод `show`.

Проверьте вашу реализацию при помощи примера использования. А также с помощью собственных примеров.