Домашнее задание к лекции 4.1 «Символы и итераторы»

Перед началом работы

1.Активируйте строгий режим соответствия.

Задача № 1. Генератор штрих-кодов

Для печати наклеек со штрих-кодами на коробки необходимо реализовать класс `BarcodeGenerator`. Экземпляр этого класса должен создавать и возвращать случайные коды вида `AA-1419`. В этом коде `AA` — префикс, все коды получают один и тот же префикс. Префикс может отсутствовать. `1419` — случайный числовой код. Количество символов в этом коде задаётся в конструкторе генератора. По умолчанию это `1`.
Также реализовать возможность смены префикса через символьное свойство. По умолчанию префикс отсутствует. В этом случае символа `-` в штрих-коде быть не должно.

Описание конструктора и экземпляра

Конструктор

Принимает один аргумент:

1.`size` — количество символов числовой части кода, число, по умолчанию равен `1`.

Экземпляр

Имеет всего один метод:

`create` — не принимает аргументов, возвращает `строку` из префикса и кода, разделенных символом `-`. Если префикса нет, то символ `-` в код не добавляется. Числовая часть должна быть случайной и иметь такое количество символов, которое было передано в конструктор генератора при создании.

Пример использования

```
const generator = new BarcodeGenerator(4);

generator[BarcodeGenerator.prefix] = 'AA';
console.log(generator.create());

generator[BarcodeGenerator.prefix] = 'XX';
console.log(generator.create());
console.log(generator.create());
console.log(generator.create());
```

```
delete generator[BarcodeGenerator.prefix];
console.log(generator.create());
```

Если все реализовано верно, вы получите такой вывод:

```
AA-1419
XX-4031
XX-1600
XX-3184
9318
```

Процесс реализации

1. Создайте класс `BarcodeGenerator`, определив конструктор.
2. Реализуйте в методе `create` генерацию случайного штрих-кода с заданным количеством символов. Пока без префикса.
3. Создайте символ для символьного свойства, в котором будет храниться префикс.
4. Используйте символьное свойство с префиксом для генерации штрих-кода.

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

Задача № 2. Электронная очередь

На некоторых планетах, где у нас есть пункт самовывоза, принята шестнадцатеричная система счисления. И поэтому порядковые номера в электронной очереди нужно выдавать в этой системе счисления.

Создайте класс `HexRange`, который реализует листаемый диапазон шестнадцатеричных чисел.

Описание конструктора и экземпляра

Конструктор

Принимает два аргумента:

1. `from` — начало диапазона, число в десятеричной системе счисления;
2. `to` — конец диапазона, число в десятеричной системе счисления.

Экземпляр

Не имеет методов. Но его можно листать с помощью `for-of` или разбирать операторами деструктуризации. Каждый элемент — это строка, шестнадцатеричное представление очередного элемента диапазона.

Пример использования

```
let queue = new HexRange(247, 253);
console.log(...queue);
```

Если все реализовано верно, вы получите такой вывод:

```
f7 f8 f9 fa fb fc fd
```

Процесс реализации

1. Создайте класс `HexRange`, описав конструктор.
2. Сделайте экземпляры этого класса итерируемыми.

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

Задача № 3. Рабочие дни

Для планирования доставки, составления графика работы менеджеров отдела продаж и для решения других задач нам нужна возможность листать только рабочие дни в заданном диапазоне. Рабочими днями у нас будут всегда дни с понедельника по пятницу включительно.

Для использования в разных модулях системы создайте класс `DateRange`. Если экземпляр этого класса листать через `for-of`, то можно получить только рабочие дни.

Описание конструктора и экземпляра

Конструктор

Принимает два аргумента:

1. `from` — дата начала диапазона, объект `Date`;
2. `to` — дата окончания диапазона, объект `Date`.

Экземпляр

Не имеет свойств и методов. Но при листании через `for-of` в каждой итерации предоставляет очередной рабочий день — объект `Date`, начиная с даты начала, если это рабочий день, и заканчивая датой окончания, если она выпадает на рабочий день.

Пример использования

```
const from = new Date(2017, 2, 13, 23, 59);
const to = new Date(2017, 2, 21, 0, 1);
```

```
let range = new DateRange(from, to);

for (let day of range) {
  console.log(day.toLocaleDateString('ru-Ru'));
}
```

Если все реализовано верно, вы получите такой вывод:

```
13.03.2017
14.03.2017
15.03.2017
16.03.2017
17.03.2017
20.03.2017
21.03.2017
```

Процесс реализации

1. Создайте класс `DateRange`, описав конструктор, принимающий два аргумента.
2. Сделайте экземпляры этого класса итерируемыми.

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