**Лаба 12**

Map – это коллекция для хранения пар «ключ-значение». Ключ любой. Ключи сохраняются без преобразования типов (в объекте ключи - строки и symbol). Проверка ключей на равенство выполняется при помощи Object.is() (в частности, NaN считается равным NaN).

– size –количество элементов

– clear() – очищает коллекцию от всех элементов

– delete(key) – пытается удалить пару по ключу key.

– forEach() – выполняет аргумент-функцию для каждой пары

– get(key) – получает элемент по ключу key (нет ключа – undefined)

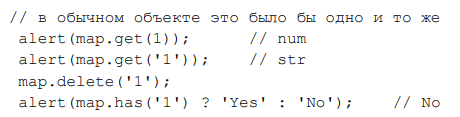
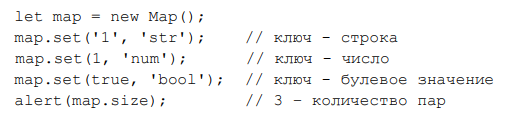
– has(key) – проверяет, есть ли ключ key (нет ключа – false)

– set(key, value) – записывает по ключу key значение value

– keys() – итератор ключей в порядке вставки пар

– values() – итератор значений в порядке вставки пар

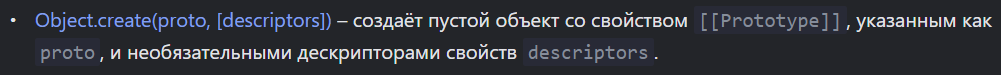
– entries() – итератор из массивов [ключ, значение] в порядке вставки пар

Метод Object.keys возвращает массив строковых элементов, соответствующих именам перечисляемых свойств, найденных непосредственно в самом объекте. Порядок свойств такой же, как и при ручном перечислении свойств в объекте через цикл. 

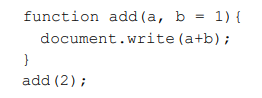
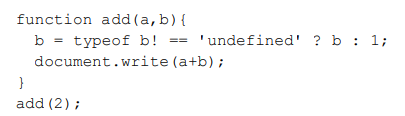
Set – множество неповторяющихся значений. ВСЕ ОСТАЛЬНОЕ ТАКОЕ ЖЕ КАК В МАP.

При повторных вызовах set.add() с одним и тем же значением ничего не происходит.

Метода Object.create - создание новых объектов. Этот метод позволяет указывать объект прототип для нового вашего объекта без определения функции конструктора.

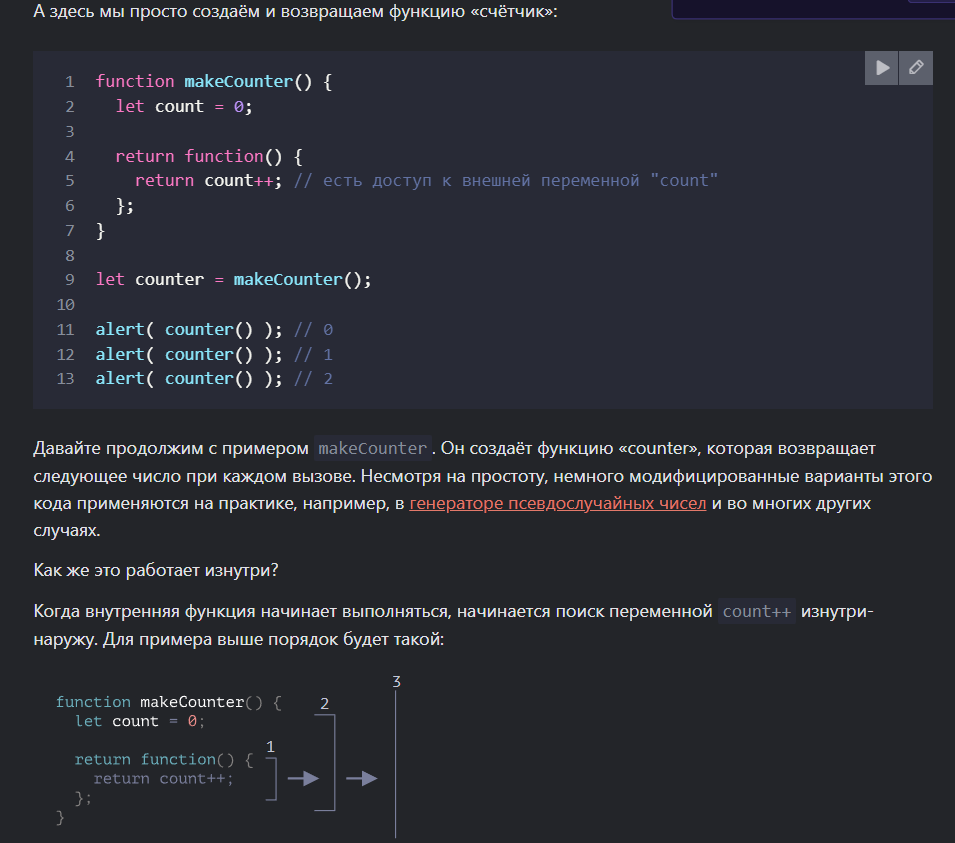


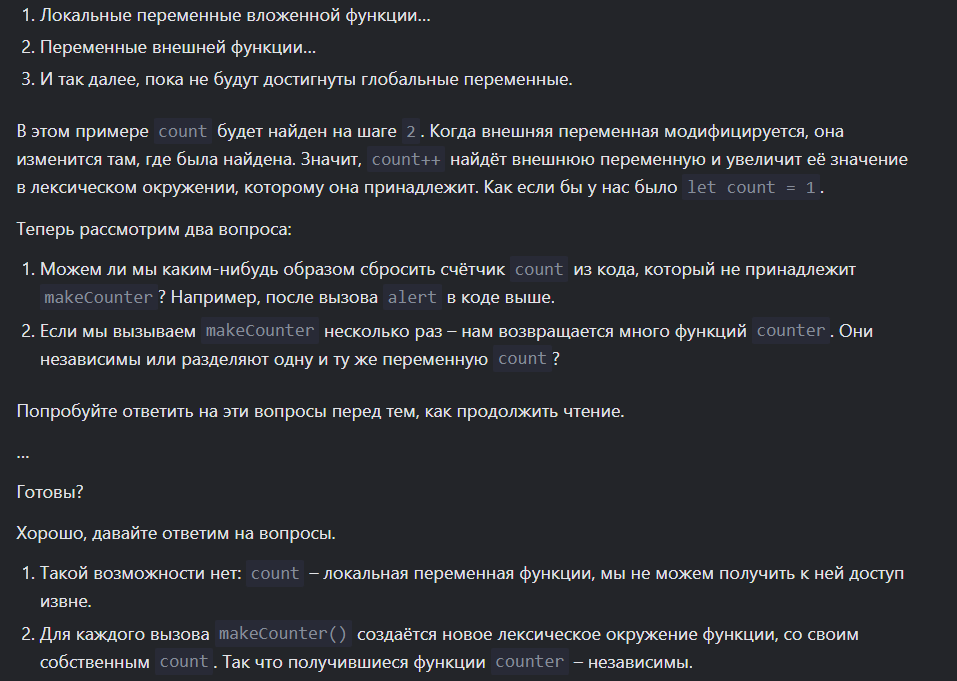
Стандартное значение параметров функций undefined. Задать иное значение по умолчанию -предназначены параметры по умолчанию. Ранее для проверки и задания стандартных значений использовалось тело функции, где параметры сравнивались с undefined. В приведённом ниже примере, для параметра b не передано значение при вызове функции, в результате он будет иметь значение undefined и результатом вычисления a+b в функции add() может быть значение NaN.



С параметрами по умолчанию проверка их значений в теле функции не требуется. Надо указать 1 в качестве значения по умолчанию для параметра b в заголовке функции.

Замыкание - если функция возвращает свою вложенную функцию g(), то все переменные и аргументы функции из scope F() (и их значения) будут доступны в g().





**Лаба 13**

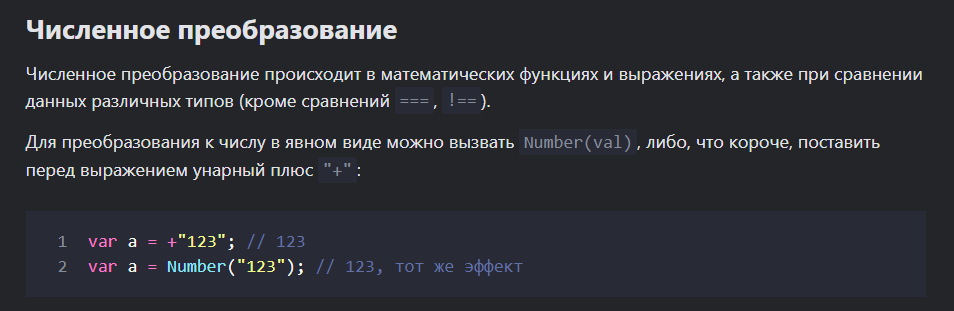
Метод includes() — это метод объектов типа Array, который определяет, есть ли в массиве искомый элемент(в отличие от indexOf, подходит и для работы со значениями NaN (выводит -1,код не работает)).

Синтаксис: arr.includes(searchElement[, fromIndex = 0])

− searchElement – искомый элемент

− fromIndex (необязательный) – позиция в массиве, с которой начинать поиск элемент searchElement.

Инфиксный оператор, который используется для возведения в степень. Такой оператор, выглядящий как «\*\*», он может служить заменой Math.pow().



**Лаба 14**

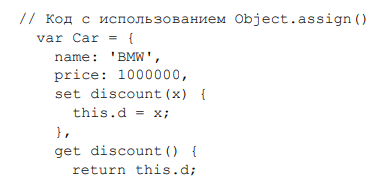
Метод Object.values() — новая функция, которая похожа на Object.keys(), но возвращает все значения собственных свойств объекта, исключая любые значения в цепочке прототипов.

Метод Object.entries() похож на метод Object.keys(), но вместо того, чтобы возвращать лишь ключи, он возвращает, в виде массива, и ключи, и значения. Это упрощает выполнение операций, предусматривающих использование объектов в циклах, или операций преобразования обычных объектов в объекты типа Map.

У объектов типа String есть два новых метода: String.prototype.padStart() и String.prototype.padEnd(). Они позволяют присоединять к строкам, в их начало или конец, некоторое количество символов для дополнения строк до заданной длины. Это удобно, если нужно выровнять текст.

Object.getOwnPropertyDescriptors()

Метод возвращает все сведения (включая данные о геттерах и сеттерах) для всех свойств заданного объекта. Главная причина добавления этого метода заключается в том, чтобы позволить создавать мелкие копии объектов и клонировать объекты, создавая новые объекты, при этом копируя, помимо прочего, геттеры и сеттеры. Метод Object.assign() этого не умеет. Он позволяет выполнять мелкие копии объектов, но не работает с их геттерами и сеттерами. В следующих примерах показана разница между Object.assign() и Object.getOwnPropertyDescriptors(), а также продемонстрировано использование метода Object.defineProperties() для копирования исходного объекта car в новый объект, ElectricCar. Тут можно заметить, что, благодаря использованию Object.getOwnPropertyDescriptors(), функция discount(), играющая роль и геттера, и сеттера, также копируется в целевой объект.

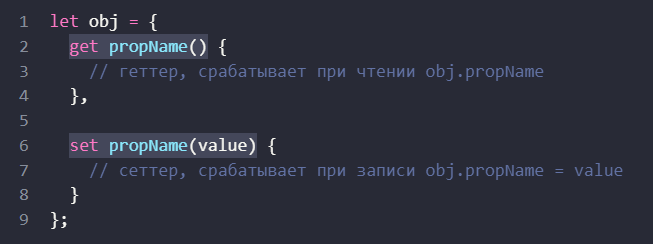


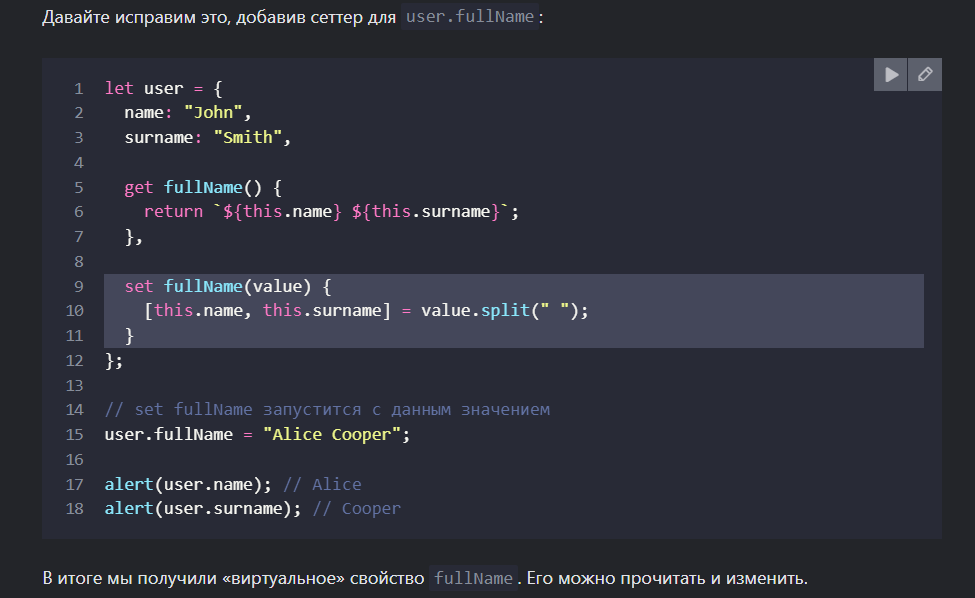


Есть два типа свойств объекта.

Первый тип это свойства-данные

Это свойства-аксессоры. По своей сути это функции, которые используются для присвоения и получения значения, но во внешнем коде они выглядят как обычные свойства объекта.

Свойства-аксессоры представлены методами: «геттер» – для чтения и «сеттер» – для записи. 

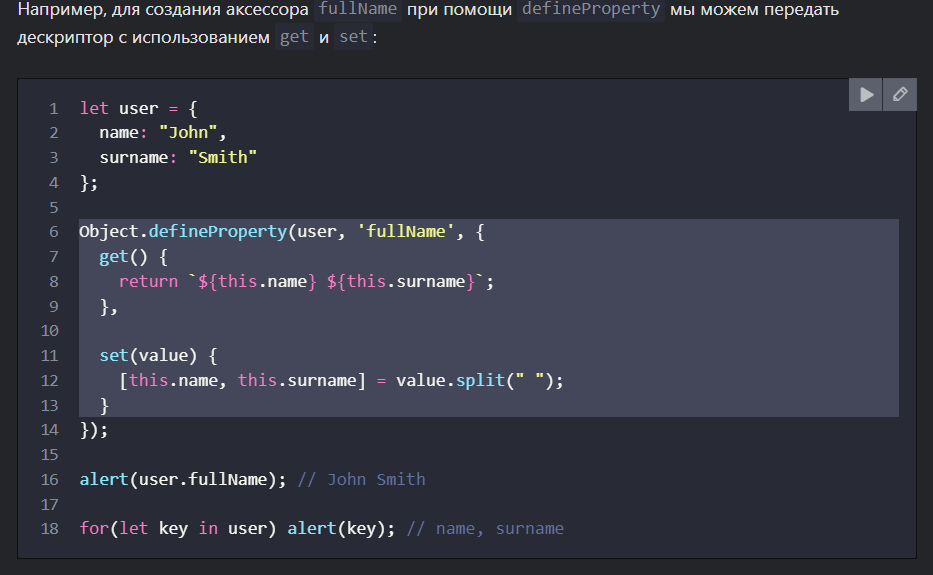


Дескрипторы свойств-аксессоров отличаются от «обычных» свойств-данных.

Свойства-аксессоры не имеют value и writable, но взамен предлагают функции get и set.

То есть, дескриптор аксессора может иметь:

* **get** – функция без аргументов, которая сработает при чтении свойства,
* **set** – функция, принимающая один аргумент, вызываемая при присвоении свойства,
* **enumerable** – то же самое, что и для свойств-данных,
* **configurable** – то же самое, что и для свойств-данных.



Метод **Object.defineProperty()** позволяет точно добавлять или изменять свойства объекта.

**Дескриптор данных** — это свойство, имеющее значение, которое может быть (а может и не быть) записываемым.

**Дескриптор доступа** — это свойство, описываемое геттером и сеттером. Дескриптор может быть только чем-то одним из этих двух типов; он не может быть одновременно обоими.

И дескриптор данных, и дескриптор доступа являются объектами. Они обладают следующими обязательными ключами:

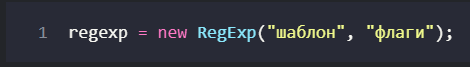
Configurable- тип этого дескриптора свойства может быть изменён и если свойство может быть удалено из содержащего его объекта.

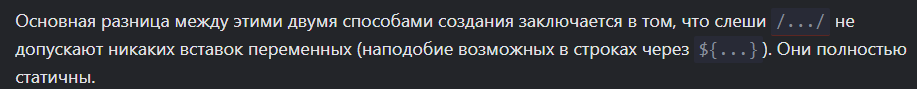
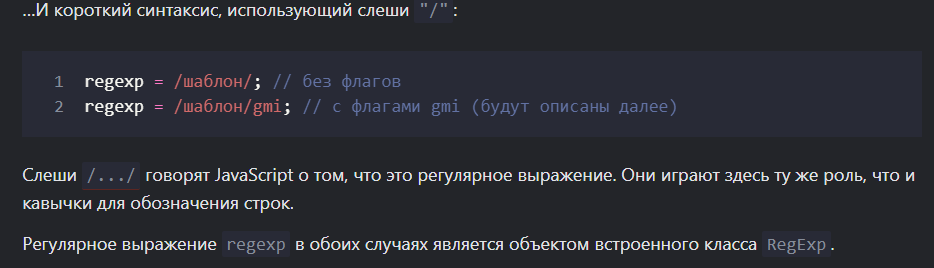
Enumerable равен true только в том случае, если это свойство можно увидеть через перечисление свойств содержащего его объекта.

//Метод Object.defineProperty() определяет новое или изменяет существующее свойство непосредственно на объекте, возвращая этот объект.

//Геттер(получить) — это метод, который получает значение определённого свойства.

//Сеттер (присвоить) — это метод, который присваивает значение определённому свойству объекта.

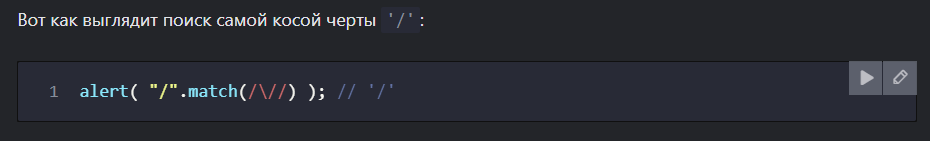
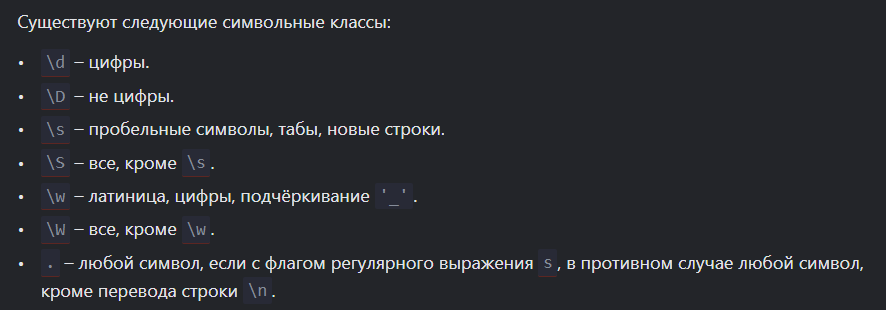
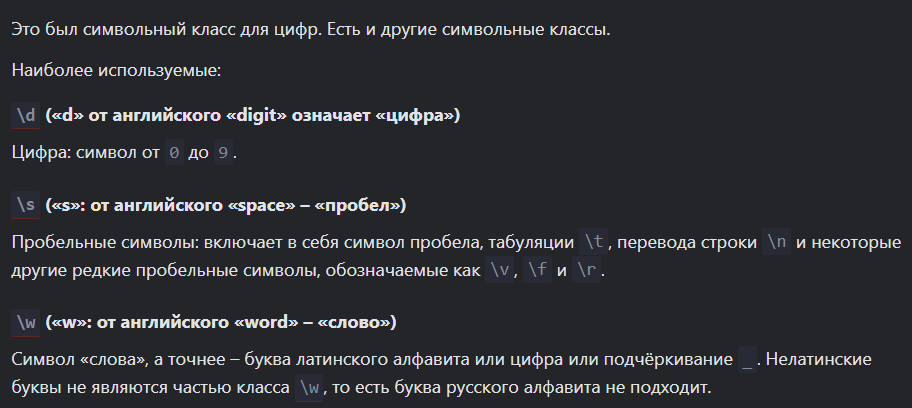
****

****

I – регистр

G – все совпадения без него только первое

m - многострочный режим



Квантификатор

