ПРОТОТИПЫ

1) Создать функцию-конструктор, принимающую в качестве параметра название самолета и возвращающую объект с 2 свойствами: 1) **name**, значение которого берется из параметра; 2) **flies**, значение которого по умолчанию для всех объектов задано как **false**.

Добавить в прототип функции-конструктора 2 метода: 1) обозначает взлет и меняет значение **flies** на **true**; 2) обозначает посадку и меняет значение **flies** на **false**.

2) Создать функцию-конструктор, принимающую 2 параметра: 1) **carModel** – название автомобиля; 2) **kmPetLiter** – количество километров, которое можно проехать на 1 литре топлива. Еще 2 свойства – **fuelTank** (топливный бак) и **odometer** (одометр) – должны задаваться как **0** по умолчанию.

Добавить в прототип функции-конструктора следующие методы:

I) **refuel (liters)** – добавляет **liters** к свойству **fuelTank**, т.е. автомобиль заправляется **liters** литрами топлива.

II) **drive** (**distance**) – увеличивает свойство **odometer** на **distance**, уменьшает свойство **fuelTank** на $\frac{\textbf{distance}}{\textbf{kmPetLiter}}$. Если **distance** оказывается больше или равно максимальному количеству км, которое может проехать автомобиль, **drive** должен вернуть строку «Закончилось топливо после преодоления *** км. На одометре *** км. Осталось *** литров топлива». Если **distance** оказывается меньше, чем максимальное количество км, которое может проехать автомобиль, **drive** должен вернуть строку «На одометре *** км, в топливном баке *** литров топлива».

3) Создать собственную реализацию метода **Object.create(prototype, [properties])**, принимающего 2 параметра: **prototype** – прототип, который будет наследоваться создаваемым объектом; **properties** – необязательный параметр, который будет передаваться в **Object.defineProperties**. Если **prototype** не является объектом или **null**, вернуть строку **'Ошибка'**.

В результате **Object.create** должен возвращать объект с внутренним свойством **[[Prototype]]**, значение которого **prototype**. Если параметр **properties** передается, и он не равен **undefined**, **Object.create** вызовет **Object.defineProperties(obj, properties),** где **obj** – объект, возвращаемый **Object.create**.

4) Усовершенствовать метод **toString** для массивов так, чтобы он работал по следующим правилам:

I) Если массив содержит только числа, **toString** должен вернуть строку, выглядящую так же, как массив:

```
[].toString() // "[]"

[5,6,7].toString() // "[5,6,7]"
```

II) Если массив содержит строки, формат такой же, как в случае I), + в строках двойные кавычки должны быть заменены на одинарные:

```
["Hello World", 77 ,"JavaScript"].toString() //
"['Hello World',77,'JavaScript']"
```

III) Обработка вложенных массивов должна быть корректной, т.е.:

```
[1,[2,3]].toString() // "[1,[2,3]]"

[1,2,[3,4],[5,6],[[7]],[8,[9]]].toString() //
"[1,2,[3,4],[5,6],[[7]],[8,[9]]]"
```

IV) Булевые значения должны обрабатываться «как есть», т.е. без кавычек и пр.

```
[true,false].toString()// "[true,false]"
```