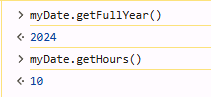
# **Прототипы и наследование**

Рассмотрим пример. В JS есть нативный(встроенный) объект Date и мы можем использовать его для работы с датами и временем. Чтобы использовать функционал объекта Date нужно сперва создать собственный экземпляр этого объекта при помощи new.

*const* myDate = new *Date*();

Далее можем использовать встроенные свойства и методы, которые живут в головном объекте date, для получения различных данных.



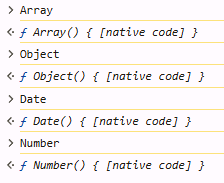
Переменная имеет тип данных ОБЪЕКТ



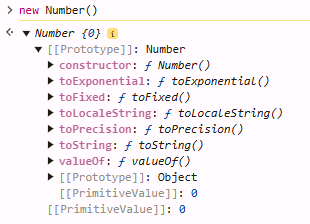
А также является экземпляром головного объекта Date. Проверим.



Кроме головного объекта Date есть и другие



На самом деле это функции и если их запускать с ключевым словом new, то будем получать объекты.



# **Создание собственного головного объекта**

Создадим головной объект при помощи функции конструктора для того, чтобы в дальнейшем создать экземпляры головного объекта.

//Функция-конструктор (головной объект)

*function* Auto(*brand*, *price*, *gas*) {

    this.brand = *brand*;

    this.price = *price*;

    this.gas = *gas*;

    drive = *function*() {

        if(this.gas > 0) {

            this.gas -= 20;

            return this.gas;

        } else {

            console.log('Бензин закончился!!!');

        }

    }

}

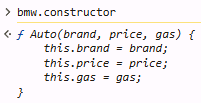
//Экземпляры головного объекта

*const* bmw = new Auto('bmw', '100,000', 100);

*const* nissan = new Auto('nissan', '40,000', 100);

console.log(bmw);

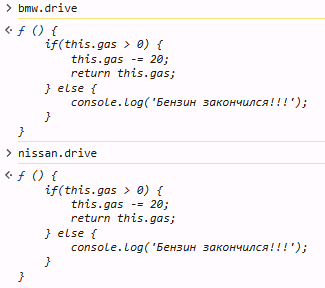
У экземпляра есть свойство constructor, которое показывает на основании какой функции был создан экземпляр.



Также можем проверить на экземпляр



Но в данной записи есть одна проблема. При создании нового экземпляра объекта мы для него создаем отдельную функцию drive()



Выглядят одинаково, но для каждого экземпляра своя функция. Когда экземпляров очень много, то подобные функции очень сильно грузят систему. Для того, чтобы написать одну функцию, и чтобы в последующем она применялась ко всем экземплярам, нужно поместить её в прототип. Таким образом мы не плодим огромное число копий функций.

//Функция-конструктор (головной объект)

*function* Auto(*brand*, *price*, *gas*) {

    this.brand = *brand*;

    this.price = *price*;

    this.gas = *gas*;

}

Auto.prototype.drive = *function*() {

    if(this.gas > 0) {

        this.gas -= 20;

        return this.gas;

    } else {

        console.log('Бензин закончился!!!');

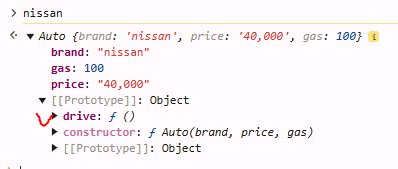
    }

}

//Экземпляры головного объекта

*const* bmw = new Auto('bmw', '100,000', 100);

*const* nissan = new Auto('nissan', '40,000', 100);



JS с начала ищет метод в свойствах объекта если не находит, то идет искать в прототипы.

В JS есть методы, которые уже помещены в прототип. Например метод массива map() помещен в прототип головного объекта Array. Создали два массива и сравнили их методы map. Они равны, следовательно данный метод берется из прототипа.

