Подводя итоги, можно смело сказать, что при использовании датчика контроля иммитансных характеристик получить экономическую выгоду в обслуживании двигателей внутреннего сгорания не составит труда.

ЛИТЕРАТУРА

1. Берко, А. В. Метод контроля моторных масел по параметрам термоокислительной стабильности и триботехническим характеристикам: Дис. Док.тех.наук: 05.11.13 [Место защиты ТПУ] – 2015. – 164 с.

2. Батурля, И.В. Диэлектрические характеристики моторных масел для силовых агрегатов, измеряемые емкостными датчиками / И. В. Батурля, А. И. Кузьмич, В. В. Баранов, В. А. Петрович, В. Ю. Серенков, С. А. Завацкий, Н. К. Фоменко, Н. С. Ковальчук // Доклады БГУИР. – 2016 – № 3 (97). – С. 103-106.

УДК 003.26

Н.А. Жиляк, доц., канд. техн. наук
(БГТУ, г. Минск)

## ОБЪЕКТЫ JAVASCRIPT. ТОНКОСТИ СОЗДАНИЯ

JavaScript является кроссбраузерным и функционирует во всех операционных системах (windows, mac os и т.д.). В отличие от объектно-ориентированных языков программирования, в JavaScript реализация объектов значительно отличается от привычного функционала и вариаций использования экземпляров, например, в C#.

Поэтому в данном материале будут описаны отличительные черты скриптовых объектов, также, какими способами их можно создать, обновить и удалить. Также будет затронута тема, касающаяся свойств, методов и конструкторов, команд и конечно же немного тема наследования. Что из себя представляет объект в JavaScript и какими возможностями обладает.

В JavaScript объектами являются простые ассоциативные массивы (их также называют хэшами).

Что же такое ассоциативный массив. Это структура данных, в которой хранится какое-то количество информации, относящееся и описывающее определенный элемент. Все данные структурированы и связаны между собой как «ключ =>значение».

К примеру, вам необходимо описать автомобили. Тогда вы создаете объект avto и описываете в массиве его характеристики. При

описании, например, марки машины (name), ее цвета (color) и стоимости (price), код реализации описанного задания будет следующим:

```
var avto = {
    name: "BMW 116i",
    color: "black",
    price: 588000};
```

Представлен один из способов создания объекта с именем «avto». Name, color и price являются ключами, по которым в ходе написания приложения можно будет обращаться.

Создать объект можно несколькими способами:

```
var avto = {}; или var avto = new Object ();
```

В обоих случаях создается пустой объект с известным именем, но первым вариантом пользуются гораздо чаще, так как его короче и удобнее писать.

Опишем теперь понятие свойство в объекте (рис. 1).

При необходимости заполнения пустого объекта параметрами необходимо добавить свойства, которые также выше были названы ключами. И опять-таки существует два способа объявления свойств.

Необходимо отметить, что в JavaScript нет строгих рамок по созданию и инициализации таких параметров. Новые свойства могут появляться на протяжении всего кода, удаляться и обновляться.

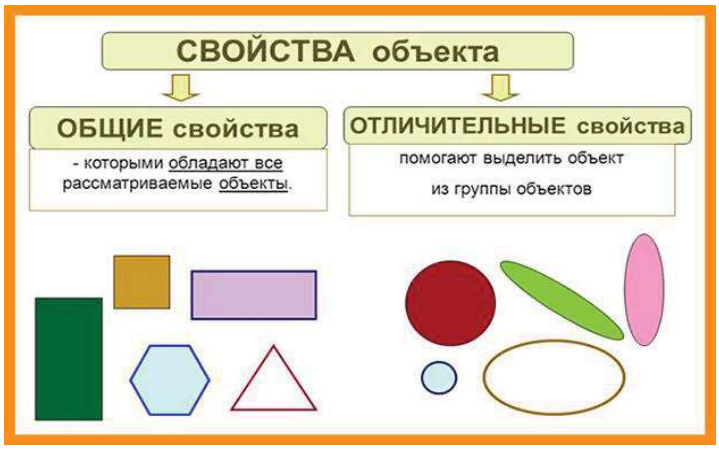


**Рисунок 1 – Представление свойств объекта**

Таким образом, можно сразу создать все ключи или же объявлять их по мере поступления.

Первый способ. Создание и обращение к свойствам через точку. Для реализации такого варианта нужно написать имя объекта, а после через точку приписать к нему наименование ключа и далее через знак равно присвоить какое-то значение:

```
avto.name = " BMW 116i"
```

А вот в таком способе к существующим ключам вы добавите еще один элемент: avto.count_door = 5. Этот способ используется то-

гда, когда имя свойства уже известно и нужно проделать определенные действия со значениями.

Второй способ. Ничем не отличается от первого, если сравнивать их предназначения. Однако этот способ обладает небольшим преимуществом. Для такого варианта используются квадратные скобки:

```
avto[“name”] = “ BMW 116i”
```

А приятным добавлением является возможность создавать наименование свойств в виде любой строки. К примеру,

```
avto[“name of the car”] = “ BMW 116i”
```

Работа с ключами через квадратные скобки используется тогда, когда какие-то параметры вводятся пользователем и хранятся в переменных или, когда заранее неизвестны названия свойств. Например, пользователь запрашивает стоимость выбранного автомобиля. В переменную записывается элемент, который был вызван, а в ответ передается цена:

```
var avto = {};
avto.name = "BMW_116i";
avto.price = 588000;
var key = 'price'; // была запрошена цена машины
          alert( avto[key] );
```

Для удаления свойств используется команда delete. Так, если к последнему примеру снизу дописать вот такие две строки:

```
delete avto.price;
alert ( avto[key] );
```

Тогда с вызовом alert во второй раз диалоговое окно выдаст «undefined».

На текущем этапе сделано описание, как создать объект и описать его свойства.

Далее опишем, как создавать сразу набор одинаковых объектов и там обязательно пригодиться именно такое представление информации. В JavaScript можно быстро перебрать созданные свойства. Для этого был предусмотрен специальный механизм, известный как цикл.

В JavaScript он напоминает своим внешним видом цикл for each из языка C#. Ознакомьтесь с общим видом конструкции:

`for (var obj in object)` {// выполнение перебора}, где obj отвечает за название перечисляемых ключей, object – за их значения.

А теперь представим конкретный пример.

```
var avto = {
  name: "BMW 116i",
  color: "black",
  price: 588000
};
for (var obj in object) {
```

```
alert(obj + ':' + object[obj])
      }
```

Если говорить о создании методов в скриптовом языке, то это абсолютно простой механизм. С помощью данного механизма в любое время к любому объекту можно дописать метод или методы, которые расширяют возможности созданных ассоциативных массивов. Их также называют свойствами-функциями.

JS сам по себе очень динамичный язык.

Так, для создания метода, нужно объявить объект, а после начать писать команду, точь-в-точь напоминающую создание свойств. Однако после знака «=» пишется уже не значение, а ключевое слово function (переменная). А далее в фигурных скобках ведется перечисление действий.

Вот реализация данного механизма:

```
var avto ={}
avto.name = "BMV"
avto.year = 1999
avto.drive = function(k) { alert(«Автомобиль
проехал»+n+« км. »)}
avto.drive(300)
avto.drive(450)
```

Как видите, этот пример содержит свойства и методы, вызов которых изначально идентичен.

В JavaScript есть еще и конструкторы. В этом языке все, что использует ключевое слово «new», автоматически становится конструктором. Так, выше вы видели объявление пустого объекта в виде: avto = new Object ();. Это и есть конструктор.

Для наглядности рассмотрите представленные строки ниже.

```
var bob = new Object ();
bob.name = «Bob Smith»;
bob.age = 20;
```

Однако это не весь арсенал возможностей. В JavaScript можно создавать свои собственные конструкторы и после использовать их для объявления новых объектов. Вдобавок к этому внутри конструктора можно создавать методы.

Обычно во многих языках наследование основывается на классах, которые могут наследовать друг друга. Тогда можно услышать такие выражения, как «класс-предок», «дочерний класс» и т.д.

Однако в JavaScript все иначе. Здесь наследуются объекты.

Все наследование основывается на внутренней ссылке между объектами, которая известна под именем «прототип». Если к методу приписать через точку «.prototype», а далее прописать имя прототипа, то все объекты выбранного метода будут наследоваться от этого прототипа.