

# ООП и JavaScript

---

**JS**  
**COURSE**  
**ORT DNIPRO**

---

**ORT****DNIPRO**.ORG/**JS**

# 1. Объекты

# Объекты в JavaScript

```
2
3   let person = {
4       name: "Jhon",
5       lastName: "Smith",
6       sayHello: function(){
7           return `Hello my name is ${this.name} ${this.lastName}`;
8       }
9   }
10
11  console.log( person.sayHello() );
```

**Объект** в JavaScript представляет собой ассоциативный массив содержащий данные (свойства) и функции (методы) которые эти данные обрабатывают. **Объект** в JavaScript один из шести базовых типов данных.

# Ключевое слово **this**

```
2
3   let person = {
4       name: "Jhon",
5       lastName: "Smith",
6       sayHello: function(){
7           return `Hello my name is ${this.name} ${this.lastName}`;
8       }
9   }
10
11 console.log( person.sayHello() );
```

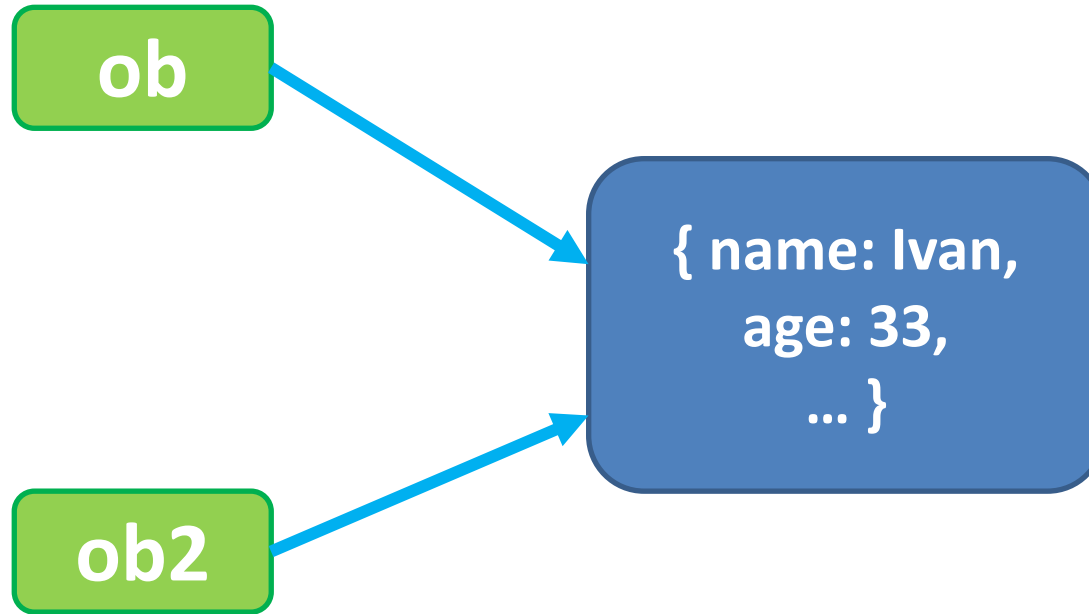
Ключевое слово **this** – ссылка на сам объект. Другими словами **this** указывает на тот ассоциативный массив (объект) которому принадлежит функция, в которой **this** используется встречается. **this** используется только в функциях объекта. **Важно: у arrow-функций нет своего this.**

# Объекты в JavaScript

```
2
3   let person = {
4       name: "Jhon",
5       lastName: "Smith"
6   }
7
8   person = null;
9
10  console.log(person, typeof person);
11
```

**null** – заглушка на случай “когда объекта нет”.

# Объекты в JavaScript



**object** - ссылочная структура данных, т.е сам объект находится где-то в памяти, а в переменной находится только ссылка на него, поэтому когда мы копируем такую переменную в другую, то копируются только ссылки, а сам объект остаётся одним и тем же.

# this привязывается в динамике

```
2
3  let func = function(){
4      return `Hello my name is ${this.name} ${this.lastName}`;
5  }
6
7  let person_1 = {
8      name: "Jhon",
9      lastName: "Smith",
10     sayHello: func
11 }
12
13 let person_2 = {
14     name: "Alice",
15     lastName: "Gates",
16     sayHello: func
17 }
18
19 console.log( person_1.sayHello() );
20 console.log( person_2.sayHello() );
21
```

**this** привязывается к объекту в момент вызова метода, поэтому одна и та же функция может входить в состав двух и большего количества объектов.

Подробнее: <https://learn.javascript.ru/object-methods>

# Конструктор – Когда нужно много однотипных объектов

```
2
3   let func = function(){
4       |   return `Hello my name is ${this.name} ${this.lastName}`;
5   }
6
7   function Person(name, lastName){
8       |   this.name      = name;
9       |   this.lastName  = lastName;
10      |   this.sayHello   = func;
11  }
12
13  let person_1 = new Person('Jhon', 'Smith');
14  let person_2 = new Person('Alice', 'Gates');
15  let person_3 = new Person('Bill', 'Roberts');
16
17  console.log(person_1.sayHello());
18  console.log(person_2.sayHello());
19  console.log(person_2.sayHello());
20
```

**Функция-конструктор** - позволяет создавать много однотипных объектов. Функция конструктор всегда должна использоваться с оператором **new**, иначе у неё не будет доступа к **this** нового созданного объекта.

Использовать оператор **return** не нужно. Конструктор может (и как правило должен) иметь параметры.

Подробнее: <https://learn.javascript.ru/constructor-new>



## 2. Прототипы

# Прототипы

У объекта может быть объект-предок, в **JavaScript** его называют **прототипом**. Если требуемое свойство (или метод) не найден в объекте, то оно ищется у **прототипа**.

**Прототип** это объект который «дополняет» своими свойствами и методами другой (дочерний) объект. Установить кто у объекта будет **прототипом** можно при помощи свойства **\_\_proto\_\_**.

Благодаря **прототипам** в **JavaScript** можно организовать объекты в «**цепочки**» так, чтобы свойство, не найденное в одном объекте, автоматически искалось бы в другом (родительском).

Подробнее: <https://learn.javascript.ru/prototypes>

# Прототипы

```
2
3   let func = function(){
4       return `Hello my name is ${this.name} ${this.lastName}`;
5   }
6
7   let family = {
8       lastName: "Smith",
9       sayHello: func
10  }
11
12  function Person(name){
13      this.name      = name;
14      this.__proto__ = family;
15  }
16
17  let person_1 = new Person('Jhon');
18  let person_2 = new Person('Alice');
19  let person_3 = new Person('Bill');
20
21  console.log(person_1.sayHello());
22  console.log(person_2.sayHello());
23  console.log(person_2.sayHello());
24
```

Свойство или метод не найденные в объекте – будут взяты из **прототипа** (или *прототипа* *прототипа*, если в цепочке прототипов искомое свойство или метод есть).

Подробнее: <https://learn.javascript.ru/prototypes>

# 3. Методы

`.toString()` / `.valueOf()`

# Методы `.toString()` / `.valueOf()` у объектов

```
2
3   let auto_1 = {
4       title: "Ford Focus",
5       id: "AE5589BH"
6   }
7
8   let auto_2 = {
9       title: "Honda Accord",
10      id: "CH5633TB",
11      toString: function(){
12          return `${this.title} (${this.id})`;
13      }
14  }
15
16  alert(auto_1);
17  alert(auto_2);
18
```

localhost:5000 says  
[object Object]

OK

localhost:5000 says  
Honda Accord (CH5633TB)

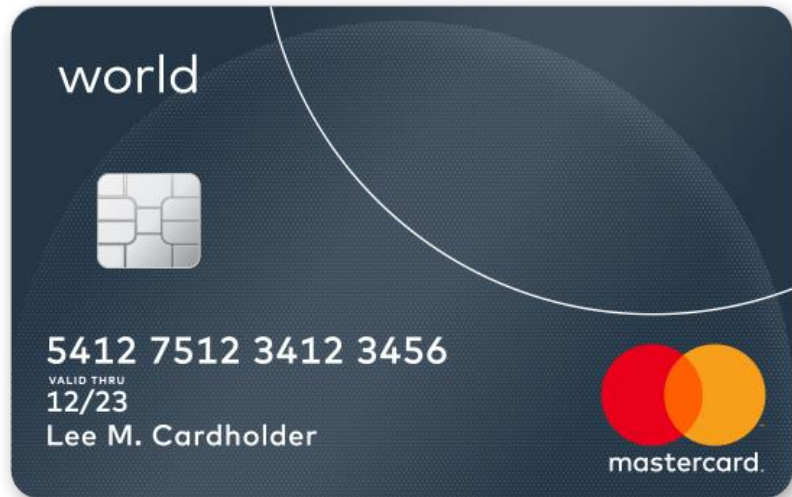
OK

Метод `.toString()`, если он определен у объекта – позволяет браузеру корректно преобразовать объект к строке. Также есть метод `.valueOf()` для преобразования к числу.

Подробнее: <https://learn.javascript.ru/object-toprimitive>

## 4. Немного практики

# Алгоритм Луна



**VISA** 4916 5526 5398 1949








5357 6872 3409 1447

*Алгоритм Луна проверяет контрольную сумму числа, широко применяется для проверки корректности номера банковских карт.*

**Задача:** пользователь вводит номер банковской карты, необходимо проверить не ошибся ли он.

Подробнее: [https://uk.wikipedia.org/wiki/Алгоритм Луна](https://uk.wikipedia.org/wiki/Алгоритм_Луна)

# Генератор номера карты

 Visa	 MasterCard	 Discover	 AmericanExpress	 JCB
✓ 4412530595659632	✓ 5287324989755118	✓ 6011139619422678	✓ 340849911182813	✓ 3539584124038594
✓ 4813431262431071	✓ 5369658110635785	✓ 6011117040432748	✓ 345673843441369	✓ 3588422734539547
✓ 4381493988886337	✓ 5153000135610537	✓ 6011406220044898	✓ 345616475358716	✓ 3538044621974255
✓ 4739306813042299	✓ 5327520507510974	✓ 6011774117039986	✓ 375103335418603	✓ 3528852467705472
✓ 4464941706819170	✓ 5155034861872910	✓ 6011069037122495	✓ 375423401400255	✓ 3579534744222947
<a href="#">Generate Visa ➤</a>	<a href="#">Generate MasterCard ➤</a>	<a href="#">Generate Discover ➤</a>	<a href="#">Generate AmEx ➤</a>	<a href="#">Generate JCB ➤</a>

Генератор номеров банковских карт:

<https://www.freeformatter.com/credit-card-number-generator-validator.html>



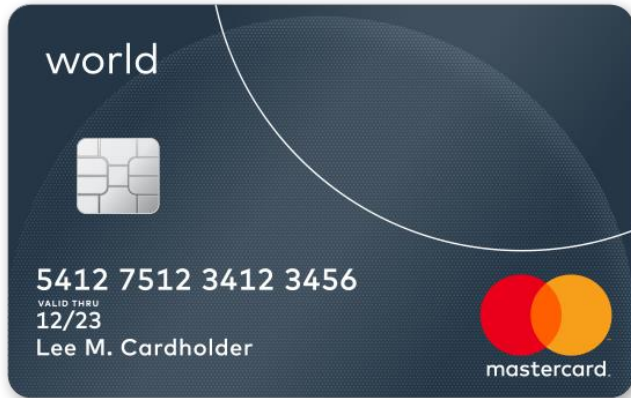
**На следующем занятии**

**На следующем занятии**

**Принципы и подходы ООП в JavaScript и  
всё, что с этим связано... часть 2**

**Домашнее задание**  
**/сделать**

# Домашнее задание #Е.1 | «Проверка номера карты»



**Задача:** Пользователь вводит номер банковской карты, необходимо проверить **корректный он или нет**. И определить тип платёжной системы: **Visa, Mastercard или Другая**.



**Подсказка:**

MasterCard это не только 5-ка в начале,  
Длинна номера карты это не всегда 16 цифр,  
Генератор номеров вам в помощь.