

Объекты и принципы ООП в 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.

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

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.**

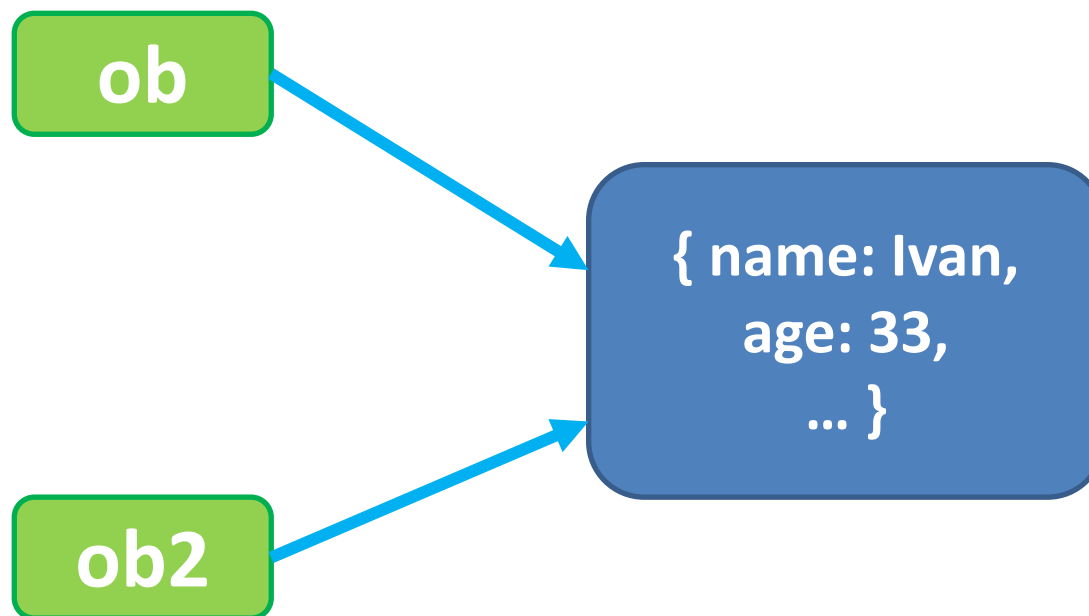
Подробнее: <https://learn.javascript.ru/object-methods>

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. Классы

Классы в ECMAScript 2015-2019

Классы пришли в JavaScript из других (типизированных) языков программирования. В которых классы применяли для описание структуры объектов которые на основе класса создаются. **Класс** выступают своего рода «чертежом» по которому будут создаваться объекты.

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

```
class Parcel{
  #code;
  #width;
  #length;
  #height;

  constructor(code, w, l, h){
    this.#code = code;
    this.#width = w;
    this.#length = l;
    this.#height = h;
  }

  getVolume(){
    return this.#width * this.#length * this.#height;
  }

  getReport(){
    return `Parcel ${this.code}: ${this.getVolume()}`;
  }
}

let box = new Parcel(100, 20, 45);

console.log(box.getReport());
```

Классы в ECMAScript 2015-2019

Классы в JavaScript'е являются лишь надстройкой («маскировкой», «синтаксическим сахаром») над **прототипной** моделью построения объектов. И не являются её заменой.

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

Классы в ECMAScript 2015-2019

По сути описывая **класс** мы создаём функцию **конструктор** в которой идёт перечисление свойств и методом будущего объекта. А далее эта функция вызывается через оператор **new**.

В **ES2019** была добавлена возможность создавать **приватные** (закрытые) свойства и методы. К этим методам есть возможность обратиться только из методов объекта. Из вне они недоступны.

Их легко отличить по символу **#** в начале имени.

Подробнее: <https://learn.javascript.ru/private-protected-properties-methods>

4. 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>

5. Обект Date

Дата/Время в JavaScript

```
2
3   let currentDate = new Date();
4   console.log(currentDate);
5   console.log(currentDate.toISOString());
6
7   let dateA = new Date(2019, 10, 18, 17, 23, 56);
8
9   console.log(dateA, +dateA);
10
```

В JavaScript есть (*относительно*) удобные возможности работы с датой и временем – объект **Date**. Дату можно преобразовать к **UTC**-виду или **timestamp**'у, и получить отдельные её компоненты (*год, месяц, ... минуты, секунды*).

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

Дата/Время в JavaScript

```
2
3   let newYear2020 = new Date(2020, 0, 1, 0,0,0);
4   let now          = new Date();
5
6   let diff = newYear2020 - now;
7
8   diff = Math.floor(diff / (1000 * 60 * 60 * 24));
9
10  console.log(`New Year 2020 after ${diff} days`);
11
```

Две даты можно вычитать одну из другой, в результате мы можем получить разницу в миллисекундах между этими датами. Это возможно за счёт преобразования даты к числу (**Timestamp'y**) которое показывает кол-во миллисекунд прошедшее от начала Unix-эпохи.

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

Дата/Время в JavaScript

Важные моменты при работе с **датой/временем**:

- 1) Не забывать про разницу между местным и UTC-временем;
- 2) Не забывать про смещение (метод: **.getTimezoneOffset()**);
- 3) Помнить о возможности преобразования даты времени в **timestamp** и обратно;
- 4) Помнить о возможности выполнять **вычитание** дат (и тем самым находить продолжительность какого-либо процесса);
- 5) JavaScript даёт определённые возможности по форматирование вывода даты/времени, при помощи методов **.toLocaleString()**, **.toLocaleDateString()**, **.toLocaleTimeString()**. Но эти возможности крайне ограничены.

Подробнее: <https://habr.com/ru/company/mailru/blog/438286/>

6. Глобальный объект `globalThis` (`window`)

Глобальный объект **window** (**globalThis**)

Браузер добавляет в JavaScript всего один объект – **window (globalThis)**. Но этот объект содержит все необходимые инструменты для манипуляции HTML-документом.

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

Глобальный объект **window**

Объект **window** можно использовать неявно, т.е. опускать его имя при написании кода.

Свойства и методы window

```
.setInterval() ;  
.setTimeout() ;  
.alert() ;  
.prompt() ;  
.confirm() ;  
...
```


7. Множество (Set)

Множество / Set

```
2
3  let set = new Set();
4
5  set.add("Jhon");
6  set.add("Helen");
7  set.add("Jhon");
8  set.add("Maria");
9  set.add("Jane");
10 set.add("Bill");
11
12 console.log(set); // {"Jhon", "Helen", "Maria", "Jane", "Bill"}
13
14 console.log( set.has('Maria') ); //true;
15 console.log( set.has('Samuel') ); //false
16
17 set.delete("Jane");
18 console.log(set); //{"Jhon", "Helen", "Maria", "Bill"}
19
```

Set – коллекция без ключей (создаётся при помощи ключевого слова **new**), позволяет хранить любые типы данных. Элемент множества встречаться в нём не более чем один раз. Есть возможность узнать есть ли элемент во множестве (метод **.has(...)**), а также узнать размер множества (свойство **.size**).

Избавление от дубликатов при помощи Set

```
2
3 let arr = ['Jane', 'Jhon', 'Maria', 'Alice', 'Jane', 'Peter', 'Alice', 'Donald'];
4
5 console.log(arr); //["Jane", "Jhon", "Maria", "Alice", "Jane", "Peter", "Alice", "Donald"]
6 console.log("Array length:", arr.length); //8
7
8 let set = new Set(arr);
9
10 console.log(set); // {"Jane", "Jhon", "Maria", "Alice", "Peter", "Donald"}
11 console.log(set.size); //6
12
13 let new_arr = Array.from(set);
14
15 console.log(new_arr); //["Jane", "Jhon", "Maria", "Alice", "Peter", "Donald"]
16 console.log("New array length:", new_arr.length); //6
17
```

8. Принципы модульного тестирования (Unit Testing)

Unit testing – модульное тестирование

```
2
3  function calc_sum(a, b){
4      let result = a + b;
5      return result;
6  }
7
8  (function(){
9      let control = calc_sum(2, 3);
10
11     if(control === 5){
12         console.log("calc_sum() - OK");
13     }else{
14         console.log("calc_sum() - FAIL");
15     }
16 })();
17
```

Идея **модульного тестирования (Unit testing)** в том, чтобы писать код который будет проверять работу основного кода. Функция, как пример модуля, может быть протестирована другой, написанной нами функцией. Основная польза модульного тестирования в том, что при изменении кода функции мы может оперативно определить не поломался ли её функционал.

Unit testing – модульное тестирование

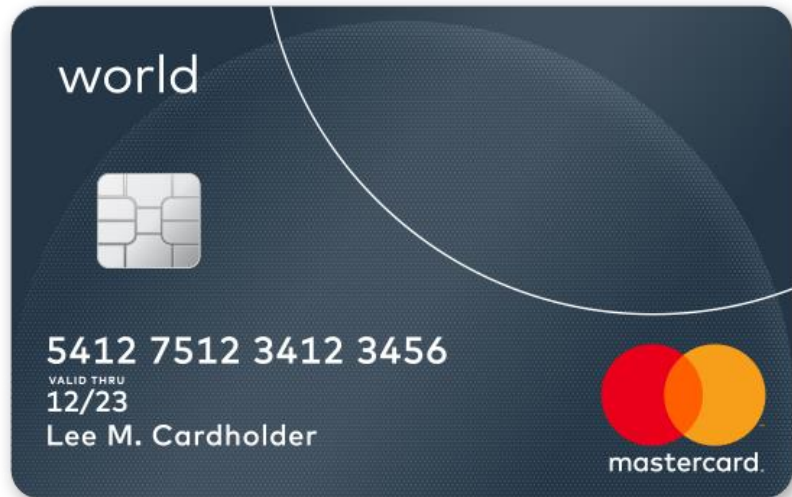
```
2
3     function calc_sum(a, b){
4         let result = a + b+1;
5         return result;
6     }
7
8     (function(){
9         let control = calc_sum(2, 3);
10
11         console.assert(control === 5, "TEST: calc_sum(2,3)");
12
13     })();
14
```

Метод **console.assert()** – удобный способ добавить вывод информации об ошибках в консоль разработчика.

Подробнее: <https://developer.mozilla.org/ru/docs/Web/API/Console/assert>

9. 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

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
Generate Visa ➤	Generate MasterCard ➤	Generate Discover ➤	Generate AmEx ➤	Generate JCB ➤

Генератор номеров банковских карт:

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

На следующем занятии

На следующем занятии

Работа с DOM

(работа с разметкой документа)

Домашнее задание
/сделать


Домашнее задание #С.1

«Проверка ИНН»

КАРТКА
фізичної особи - платника податків

повідомляє, що Григорук Володимир Григорович

одержав(ла) ідентифікаційний номер
наданий Державною податковою адміністрацією України
згідно з даними, заповненими ним (нею) в обліковій картці.
Дата занесення до Державного реєстру фізичних осіб - 06/02/1998
(картка видана для пред'явлення до органів державної реєстрації,
установ банків та інших).

 М.П. Григорук Володимир Григорович
ДЛЯ
ДОВІДОК
(підпис) (прізвище та ініціали посадової особи
органу Державної податкової служби)

06.02.1998
(дата видачі картки)

Для проверки:

3463463460 – пол женский, д.р. 28.10.1994;

2063463479 – пол мужской, д.р. 29.06.1956.

Пользователь вводит ИНН (физ. лица Украины). Необходимо определить: **корректен ли код, узнать дату рождения, определить пол и сколько полных лет человеку.**

Домашнее задание #С.2 | «Проверка номера карты»



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



Подсказка:

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