# Лекция 4

Объекты

#### Цели на сегодня:

- Понять, что такое объект и узнать области использования объектов;
- Научиться отличать методы и свойства объектов;
- Узнать, как получать, добавлять, изменять и удалять значения;
- Разобрать написание небольшой консольной игры.

### Типы данных (Data Types)

#### 1. Примитивы

- a. Числа (Numbers);
- b. Строки (Strings);
- с. Логические (Boolean)
- d. Символ (Symbol);
- e. null;
- f. undefined.

#### 2. Сложные

- а. Объекты (Objects);
- b. Maccивы (Arrays);
- с. Функции (Functions).

### NB!

```
console.log(typeof null) // object
console.log(typeof {name: 'Peter'}) // object
```

#### Объект –

это сложный тип данных, который способен хранить любые данные в формате "ключ - значение". При этом по ключу всегда можно получить доступ к значению.

Для чего используются объекты?

Вариантов использования этого типа данных огромное множество. Некоторые из них:

- 1) Словари (например: английское слово русское слово);
- 2) **Хранение информации про пост, товар, сообщение в чате** (каждый объект товара содержит название, цену, артикул);
- 3) Записи в базе данных (NoSQL решения);
- 4) **Конфигурационные файлы** (объект содержит настройки приложения и подключается как модуль);
- 5) **Коллекции** (например: объект Math, объект Console, объект Date, коллекция всех тегов на странице).

Как создать объект?

#### Создавать объекты можно по-разному:

- 1) Фигурные скобки (литеральная нотация);
- 2) Функция-конструктор;
- 3) new Object();
- 4) Object.create().

```
// Создаем пустой объект
const obj = {};
console.info(obj) // Object { }
// Второй синтаксис менее лаконичен
const secondObj = new Object();
console.info(secondObj); // Object { }
```



### Первый способ

```
const wizard = {};
wizard.name = 'Muhammad Avdol';
wizard.gender = 'Male';
wizard.power = 'Fire';
wizard.health = 200;
wizard.energy = 60;
wizard.height = 188;
wizard.weight = 90;
wizard.clothes = {
    necklace: 'Yellow rings',
    coat: '',
    pants: '',
    boots: ''
```

### Второй способ

```
const wizard = {
    name: 'Muhammad Avdol',
    gender: 'Male',
    power: 'Fire',
    health: 200,
    energy: 60,
    height: 188,
    weight: 90,
    clothes: {
        necklace: 'Yellow rings',
        coat: '',
        pants: '',
        boots: '',
    },
```

### Третий способ

ES6

```
const name = 'Muhammad Avdol';
const gender = 'Male';
const power = 'Fire';
const health = 200;
const energy = 60;
const wizard = {
    name,
    gender,
    power,
    health,
    energy,
```

Функции-конструкторы используются, когда нужно создать много похожих объектов.

В таком случае создается функция-конструктор, а объекты из нее образуют с помощью ключевого слова *new*.

```
function Knight(id, health, energy) {
    this.id = id:
    this.health = health;
    this.energy = energy;
    this.name = 'Jean Pierre Polnareff';
    this.gender = 'Male';
    this.power = 'Rapier master';
    this.height = 185;
    this.weight = 78;
    this.clothes = {
        necklace: false,
        coat: 'Fitted sleeveless black top',
        pants: 'Yellow ones',
        boots: 'Ordinary black boots',
```

```
/*
    Каждое присваивание переменной создает нового
    рыцаря, которому мы передаем id, health и energy
    Остальные параметры у них будут одинаковы.
*/
                               id, health, energy
const firstKnight = new Knight(1, 200, 120);
const secondKnight = new Knight(2, 60, 10);
```



#### Самостоятельная работа

Создать объект **room** (используя литеральную нотацию и функциюконструктор), описывающий комнату вокруг вас:

{

- 1) Стены: { количество, цвет в НЕХ },
- 2) Окна: { количество, цвет рамы, чистые или нет? },
- 3) Освещение: { количество ламп, свет включен или нет? },
- 4) Проектор: включен или нет?,
- 5) Столы: количество,
- 6) Люди: количество душ

## Методы

### Метод –

это свойство-функция в объекте.

Ключевое слово *this* (оно еще называется контекстом вызова) ссылается на текущий объект или функцию и используется для доступа к текущему объекту из метода.

```
const calc = {
    two: 2,
    addTwo: function(num) {
        return +num + this.two;
calc.addTwo(10) // 12
calc.addTwo(5) // 7
```

## ES6

```
/*
    В ES-2015 можно использовать сокращенную запись методов.
    Работает аналогично, но выглядит лаконичнее и методы теперь
    выделяются среди обычных свойств.
*/
const calc = {
    two: 2,
    addTwo(num) {
        return +num + this.two;
calc.addTwo(10) // 12
calc.addTwo(5) // 7
```

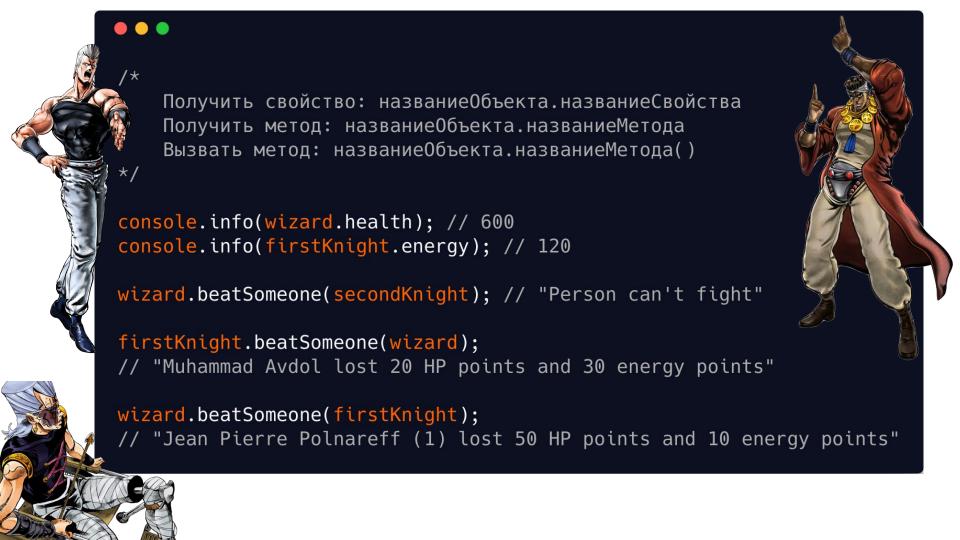
```
/*
    В функциях-конструкторах публичные методы создаются аналогично
    обычным свойствам, то есть с использованием ключевого слова
    this.
*/
const Cat = function(name) {
    this.name = name;
    this.meow = function() {
        alert('Meow!');
const Barsik = new Cat('Barsik');
Barsik.meow(); // 'Meow!'
```

```
beatSomeone(person) {
    if (person.name === 'Muhammad Avdol') {
        return 'No!';
    } else {
        person.health -= 50;
        person.energy -= 10;
    if (person.health <= 0) {</pre>
        return 'Person died.'
    if (person.energy <= 0) {</pre>
        return 'Person can\'t fight'
    return `${person.name} (${person.id}) lost 50 HP points and 10 energy points`;
```

```
this.beatSomeone = function(person) {
    if (person.name === 'Jean Pierre Polnareff') {
        return 'Nani?';
    } else {
        person.health -= 20;
        person.energy -= 30;
        if (person.health <= 0) {</pre>
            return 'Person died'
        if (person.energy <= 0) {</pre>
            return 'Person can\'t fight'
        return `${person.name} lost 20 HP points and 30 energy points`;
```

Как получить значение свойства /

вызвать метод?



Как изменять значения?

```
/*
    Для уверенности в своей победе Рыцарь (2) решил
    увеличить запас здоровья Рыцаря (1).
   Для этого он получил значение здоровья первого рыцаря
   и переписал значение на большее
*/
console.info(firstKnight.health); // 60
firstKnight.health = 300;
console.info(firstKnight.health); // 300
```

Как удалять свойства и методы?

```
/*
   Маг оказался хитрее, поэтому решил запретитьрыцарям
    сопротивляться, удалив у них метод beatSomeone
*/
delete firstKnight.beatSomeone; // true
delete secondKnight.beatSomeone; // true
  При попытке ударить мага, появится ошибка:
firstKnight.beatSomeone(wizard); // TypeError
firstKnight.beatSomeone(wizard); // TypeError
```

#### Деструктуризация

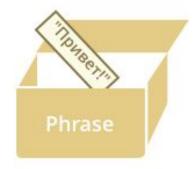


```
const wizard = {
    name: 'Muhammad Avdol',
    gender: 'Male',
    power: 'Fire',
    health: 600,
    energy: 200,
const { name, gender, power, health, energy } = wizard;
// Имя теперь можно получить из свойства объекта:
console.log(wizard.name); // "Muhammad Avdol"
// И из переменной name:
console.log(name); // "Muhammad Avdol"
```

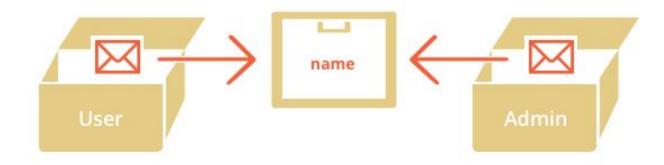
## Передача по ссылке

```
const message = 'Πρивет!';
const phrase = message;
```





```
const user = { name: 'Bacя' };
const admin = user;
```



#### Самостоятельная работа

Написать объект R2D2, в котором будут следующие методы:

- 1) Метод для сложения двух чисел (R2D2.add(a, b));
- 2) Метод для нахождения большего из двух чисел (R2D2.max(a, b));
- 3) Метод для возведения в степень (R2D2.pow (число, степень));
- 4) Метод для нахождения дискриминанта(R2D2.disk(a, b, c));
- 5) Метод для нахождения диаметра(R2D2.diam (длина));

При выполнении домашнего задания нельзя использовать готовые методы объекта Math