Объекты/ Objects

JavaScript
Courses

vk.com/js.courses

js.courses.dp.ua/files

Объекты / Objects

установить время()

звонить()

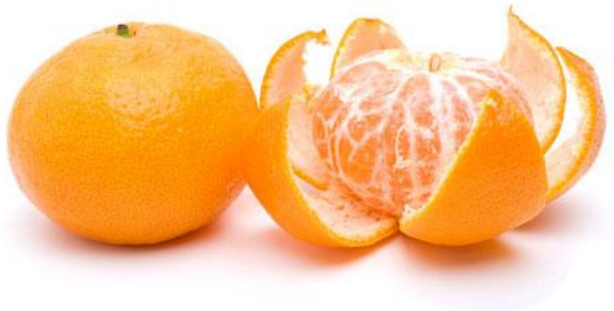
узнать время()

<< Действия()/Методы() >>

пахнуть()

почистить()

выжать сок()



циферблат: механика

вес: 300 грамм

материал: металл

<< Свойства / Аттрибуты >>

параметры

цвет: оранжевый

сахар: 15%

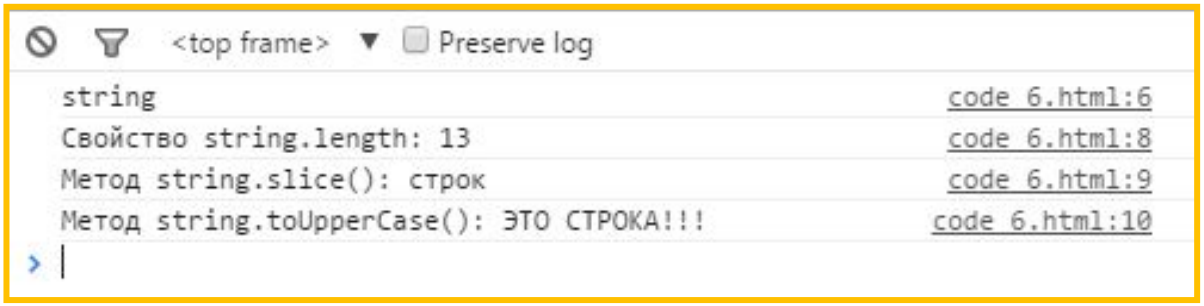
форма: шаровидная

сорт: «Дэнси»

Объекты / **Objects**

Объект в программировании — некоторая сущность, обладающая определённым состоянием и поведением, имеющая заданные значения свойств (параметров, атрибутов) и операций над ними (функций, методов).

```
3  <script>
4      var str = "Это строка!!!";
5
6      console.log(typeof(str));
7
8      console.log("Свойство string.length: " + str.length);
9      console.log("Метод string.slice(): " + str.slice(4, 9));
10     console.log("Метод string.toUpperCase(): " + str.toUpperCase());
11 </script>
```



Объекты / **Object**s

```
<script>
    var ob = {
        name: "Ivan",

        age: 19,

        say_hello: function(){
            console.log("This is" + this.name + ", age: " + this.age)
        },

        growup: function(){
            this.age++;
            console.log(this.name + " growup!");
        }
    }

    ob.say_hello();

    ob.growup();

    ob.say_hello();
</script>
```

```
This is Ivan, age: 19          xxx.html:8
Ivan growup!                   xxx.html:13
This is Ivan, age: 20          xxx.html:8
```

Объект в JavaScript представляет собой ассоциативный массив (верно и обратное утверждение) содержащий поля (свойства, элементы, такие себе переменные) с данными, и методы (функции) их обрабатывающие.

Объекты / **Object**s

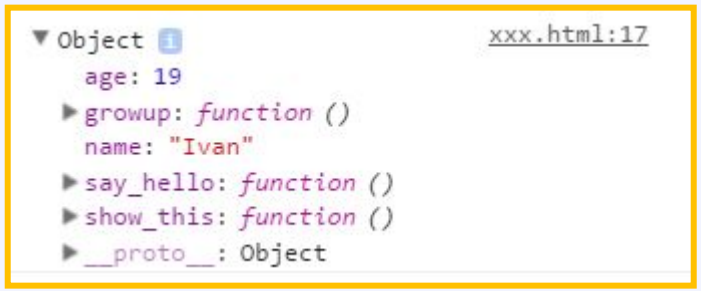
```
<script>
    var ob = {
        name: "Ivan",

        age: 19,

        say_hello: function(){
            console.log("This is " + this.name + ", age: " + this.age);
        },

        growup: function(){
            this.age++;
            console.log(this.name + " growup!");
        }
    }

    for(var i in ob){
        console.log(i, ob[i]);
    }
</script>
```

?!?

Объект в JavaScript представляет собой ассоциативный массив (верно и обратное утверждение) содержащий поля (свойства, элементы, такие себе переменные) с данными, и методы (функции) их обрабатывающие.

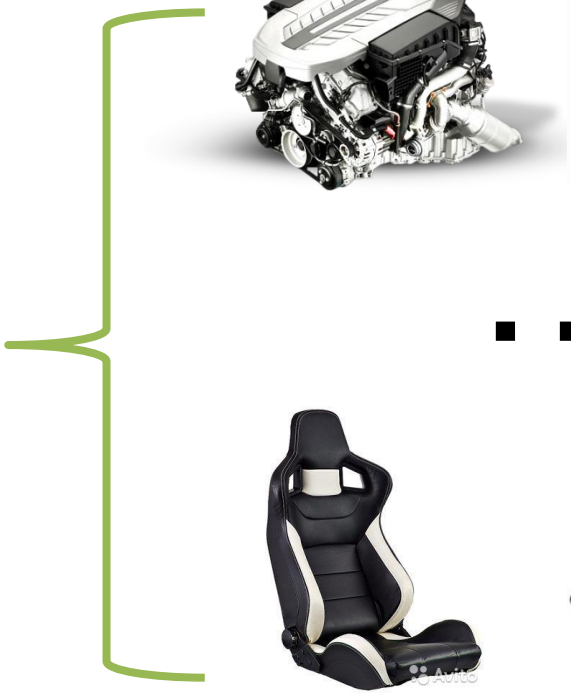
Ключевое слово **this**

```
<script>
    var ob = {
        name: "Ivan",

        age: 19,

        say_hello: function(){
            console.log("This is " + this.name + ", age: " + this.age);
        },

        growup: function(){
            this.age++;
            console.log(this.name + " growup!");
        },

        show_this(){
            console.log(this);
        }

    }

    ob.show_this();
</script>
```



*Ключевое слово **this** – ссылка на сам объект. Другими словами **this** указывает на тот ассоциативный массив (объект) которому принадлежит функция, в которой this используется встречается.*

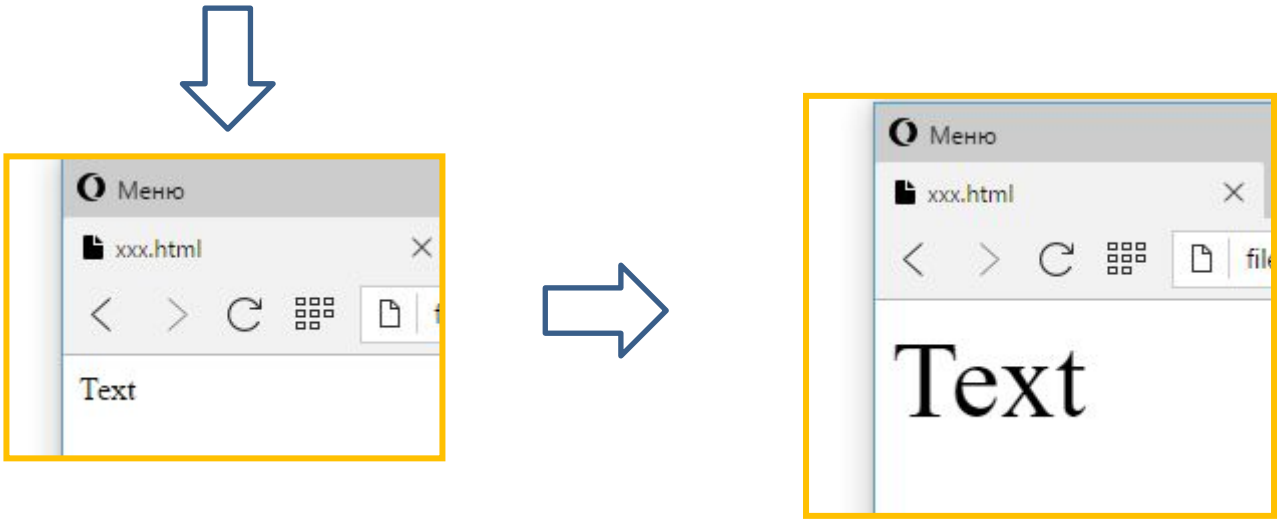
Объекты / Objects



Объект может состоять из множества других объектов. Абстракция. Принцип черного ящика.

Объекты / Objects

```
1  <html>
2  <body>
3      <div>Text</div>
4      <script>
5          var x = document.querySelector("div");
6          x.style.fontSize = 55;
7      </script>
8  </body>
9  </html>
```



Абстракция. Принцип черного ящика.

Объекты могут состоять из других объектов

```
<script>
    var ob = {
        name: "Ivan",

        address: {
            city: "Kiev",
            street: "Khreshatik",
            building: 45,
            show_adr: function(){
                console.log(this.city, this.street, this.building);
            }
        },

        say_hi: function(){
            console.log("I'm " + this.name + ", my address: " + this.address.city
            + ", " + this.address.street + ", " + this.address.building);
        }
    }

    ob.say_hi();

    console.log("---");

    ob.address.show_adr();
</script>
```

```
I'm Ivan, my address: Kiev, Khreshatik, 45    xxx.html:15
---                                           xxx.html:21
Kiev Khreshatik 45                            xxx.html:10
```

Объекты могут состоять из множества других объектов. Обращаться к ним не составляет труда. Оператор точка «.» разделяет сложенный объект и объект «контейнер». Такое разделение называют – цепочкой вызова, длинна такой цепочки ограничена только тем сколько объектов участвуют в «матрёшке».

Объекты / **Object**s

```
var element = document.getElementById("tagId");
var element_style = element.style;
element_style.color = "red";
```

```
window.document.getElementById("tagId").style.color = "red";
```

Вложенность объектов позволяет писать простой и элегантный код.

Объекты / **Objects**

```
1  <script>
2      var ob = {
3          name: "Ivan",
4          age: 32,
5          say_hello: function(){
6              console.log("It's " + this.name + ", age: " + this.age);
7          }
8      }
9
10     ob.say_hello();
11
12     ob.inc_age = function(){
13         this.age++;
14     };
15
16     ob.inc_age();
17
18     ob.say_hello();
19 </script>
```

?!?

Объекты, как и массивы – динамическая структура данных

Объекты / **Object**s

```
1  <script>
2      var ob = {
3          name: "Ivan",
4          age: 32
5      }
6
7      console.log(ob.height);
8
9      ob.say_hello();
10
11 </script>
```

?!?

Объекты, как и массивы – динамическая структура данных

Объекты / **Object**s

```
1  <script>
2      var ob = {
3          name: "Ivan",
4          age: 32,
5          say: function(){ console.log(this.name, this, age); }
6      }
7
8      console.log(typeof(ob), ob);
9
10     ob = null;
11
12     console.log(typeof(ob), ob);
13 </script>
```

```
object ▶ Object {name: "Ivan", age: 32}        xxx.html:8
object null                                     xxx.html:12
```

object* и *null

Объекты / **Objects**

```
<script>

    var a = 7;

    var b = a;

    b++;

    console.log(a, b);

</script>
```

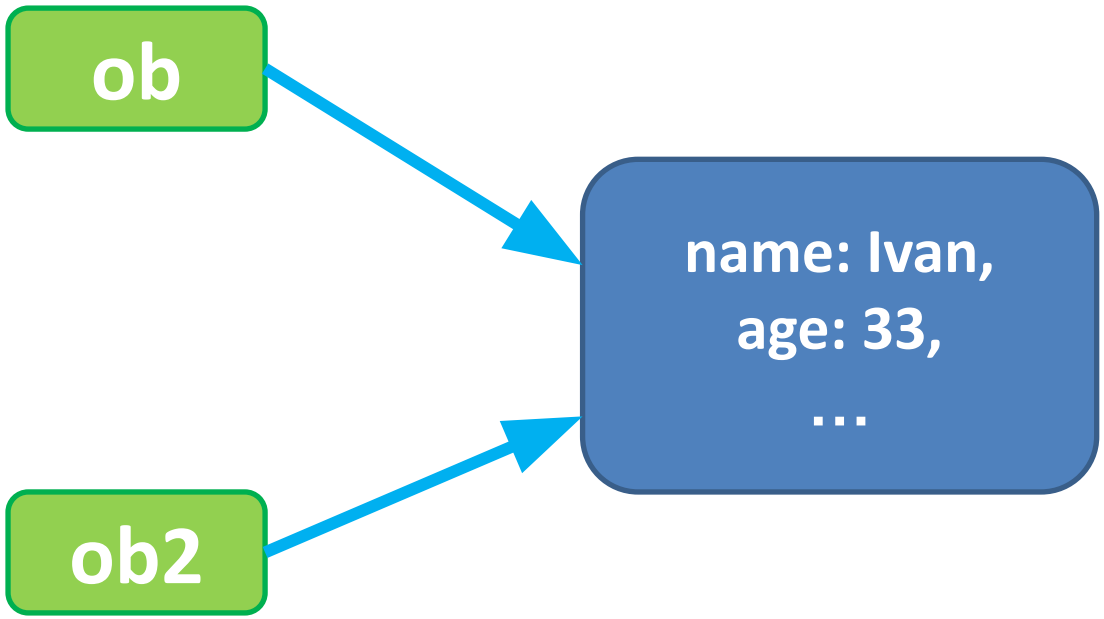
?!?

```
<script>

    var ob = {
        a: 7
    }

    var ob2 = ob;

    console.log(ob, ob2);

    ob.a++;

    console.log(ob, ob2);

</script>
```

?!?

object *- ссылочная структура данных*

Объекты / **Object**s



object *- ссылочная структура данных*

Конструктор

```
<script>
        function PersonConstructor(){
            this.name   = "Ivan";
            this.age    = 28;
            this.phone  = "+380505555555";
            this.say_hello = function(){
                alert("Person: " + this.name)
            }
        }

        var person1 = new PersonConstructor();
        var person2 = new PersonConstructor();

        console.log(person1, person2);

        person1.age = 55;

        console.log(person1, person2);
</script>
```

```
▶ PersonConstructor {name: "Ivan", age: 28, phone: "+380505555555"}        xxx.html:14
▶ PersonConstructor {name: "Ivan", age: 28, phone: "+380505555555"}
▶ PersonConstructor {name: "Ivan", age: 55, phone: "+380505555555"}        xxx.html:18
▶ PersonConstructor {name: "Ivan", age: 28, phone: "+380505555555"}
```

Функция-конструктор *- позволяет создавать много однотипных объектов.*

Конструктор

```
<script>
        function PersonConstructor(p_name, p_age, p_phone){
            this.name   = p_name;
            this.age    = p_age;
            this.phone  = p_phone;
            this.say_hello = function(){
                console.log("Person: " + this.name, "age: " + this.age, "phone: "
                + this.phone);
            }
        }

        var person1 = new PersonConstructor("Elena", 22, "0677777775");
        var person2 = new PersonConstructor("Irina", 30, "0634422354");

        person1.say_hello();
        person2.say_hello();

</script>
```

```
Person: Elena age: 22 phone: 0677777775                    xxx.html:7
Person: Irina age: 30 phone: 0634422354                    xxx.html:7
```

Функция-конструктор *- позволяет создавать много однотипных объектов. Конструктор может (и должен) иметь параметры*

Прототипы

```
1  <script>
2      var ob = {
3          name: "Elena",
4          age: 23
5      };
6
7      var ob_proto = {
8          city: "Kiev",
9          company: "Kyivstar"
10     }
11
12     console.log(ob.city);
13
14     ob.__proto__ = ob_proto;
15
16     console.log(ob.city);
17 </script>
```

```
undefined                    xxx.html:12
Kiev                         xxx.html:16
```

Прототип *– объект родитель, дополняет своими свойствами дочерний объект.*

Прототипы

```
<script>
        var company_info = {
            company_name:   "Kyivstar",
            company_adr:    "Kiev, Shevchenko street 45"
        }

        function PersonConstructor(p_name, p_age, p_phone){
            this.name   = p_name;
            this.age    = p_age;
            this.phone  = p_phone;
            this.say_hello = function(){
                console.log("Person: " + this.name, "age: " + this.age, "phone: "
                + this.phone, "company: " + this.company_name, "company adr: " +
                this.company_adr);
            }
            this.__proto__ = company_info;
        }

        var person1 = new PersonConstructor("Elena", 22, "0677777775");
        var person2 = new PersonConstructor("Irina", 30, "0634422354");

        person1.say_hello();
        person2.say_hello();

        company_info.company_name   = "McDonalds";
        company_info.company_adr    = "Kyiv, Ivanova street, 32";

        person1.say_hello();
        person2.say_hello();
</script>
```

```
Person: Elena age: 22 phone: 0677777775 company: Kyivstar company adr: Kiev, Shevchenko street 45   xxx.html:13
Person: Irina age: 30 phone: 0634422354 company: Kyivstar company adr: Kiev, Shevchenko street 45   xxx.html:13
Person: Elena age: 22 phone: 0677777775 company: McDonalds company adr: Kyiv, Ivanova street, 32    xxx.html:13
Person: Irina age: 30 phone: 0634422354 company: McDonalds company adr: Kyiv, Ivanova street, 32    xxx.html:13
```

Прототип *– объект родитель, дополняет своими свойствами дочерний объект.*

Доступ к свойствам прототипа

```
<script>
        var base_ob = {
            a: 77
        }

        var ob = {
            b: 88,
            __proto__: base_ob
        }

        console.log(ob.a, base_ob.a);

        ob.a = 99;

        console.log(ob.a, base_ob.a);

        console.log(ob, base_ob);
</script>
```

?!?

Прототип *– объект родитель, дополняет своими свойствами дочерний объект.*

Оператор **in**, прототипы и свойства объектов

```
1  <script>
2      var base_ob = {
3          a: 5
4      }
5
6      var ob = {
7          b: 10,
8          __proto__: base_ob
9      }
10
11     console.log("a" in ob);
12     console.log("b" in ob);
13     console.log("b" in base_ob);
14 </script>
```

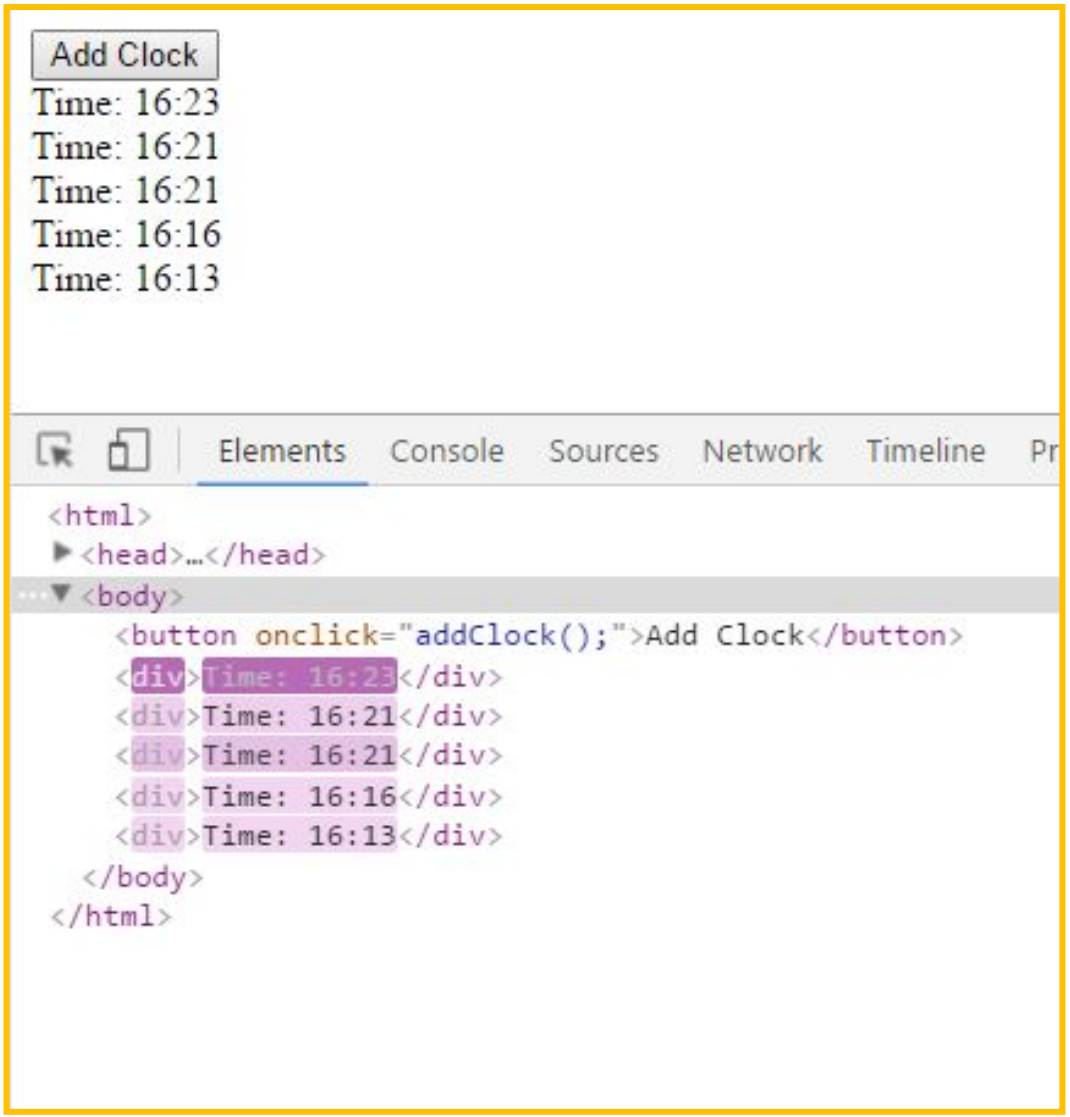
?!?

Прототип *– объект родитель, дополняет своими свойствами дочерний объект.*

Объекты «Часы»

```
<!DOCTYPE html>
<html>
<head>
<script>
    function ClockConstructor(h, m){
        this.hours      = h;
        this.minutes    = m;
        this.tag        = document.createElement("div");
        document.body.appendChild(this.tag);

        this.updateShow = function(){
            this.tag.innerHTML = "Time: " + this.hours + ":"+this.minutes;
        }

        this.tick = function(){
            this.minutes++;
            if(this.minutes > 59) {
                this.minutes = 0;
                this.hours++;
            }
            if(this.hours > 23){
                this.hours = 0;
            }
            this.updateShow();
        }

        setInterval(function(thisClock){
            thisClock.tick();
        }, 1000, this);
    }

    function addClock(){
        var clock = new ClockConstructor(15, 23);
    }
</script>
</head>
<body>
    <button onclick="addClock();">Add Clock</button>
</body>
</html>
```

Объекты «Часы»



JavaScript как язык программирования

его

концепции

Переменные / Типы / Операции

Ветвления (условные операторы)

Циклы / Массивы (структуры данных)

Функции

И

Объекты


Эрик Фримен
Элизабет Робсон
Изучаем программирование
на JavaScript
BANG!
Научись обходить
«подводные камни»
Руководство
по программированию
на JavaScript
Начни карьеру
программиста
с одной главы
Не ошибайся
в преобразованиях
типов
Тренируйся
на 120 примерах
и упражнениях
Узнай,
почему твои друзья
ничего не понимают
в функциях и объектах
O'REILLY®
ПИТЕР®