

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>
```

🔍 <top frame> ▾ Preserve log

string	code 6.html:6
Свойство string.length: 13	code 6.html:8
Метод string.slice(): строка	code 6.html:9
Метод string.toUpperCase(): ЭТО СТРОКА!!!	code 6.html:10

> |

Объекты / Objects


```
1 <script>
2   var ob = {
3     name: "Ivan",
4
5     age: 19,
6
7     say_hello: function() {
8       console.log("This is" + this.name + ", age: " + this.age)
9     },
10
11    growup: function() {
12      this.age++;
13      console.log(this.name + " growup!");
14    }
15  }
16
17  ob.say_hello();
18
19  ob.growup();
20
21  ob.say_hello();
22 </script>
```

This is Ivan, age: 19	<u>xxx.html:8</u>
Ivan growup!	<u>xxx.html:13</u>
This is Ivan, age: 20	<u>xxx.html:8</u>

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 / Objects

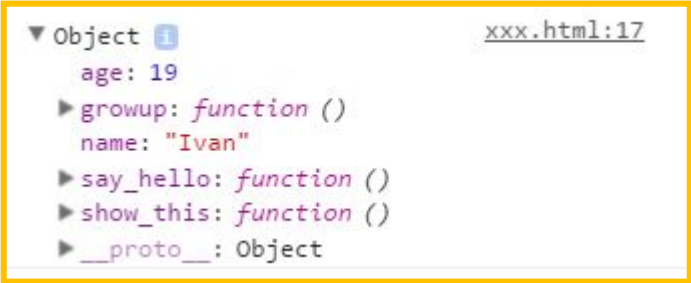
```
1 <script>
2   var ob = {
3     name: "Ivan",
4
5     age: 19,
6
7     say_hello: function(){
8       console.log("This is " + this.name + ", age: " + this.age);
9     },
10
11    growup: function(){
12      this.age++;
13      console.log(this.name + " growup!");
14    }
15  }
16
17  for(var i in ob){
18    console.log(i, ob[i]);
19  }
20 </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

```
1 <script>
2   var ob = {
3     name: "Ivan",
4
5     age: 19,
6
7     say_hello: function(){
8       console.log("This is " + this.name + ", age: " + this.age);
9     },
10
11    growup: function(){
12      this.age++;
13      console.log(this.name + " growup!");
14    },
15
16    show_this(){
17      console.log(this);
18    }
19  }
20
21  ob.show_this();
22
23 </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.

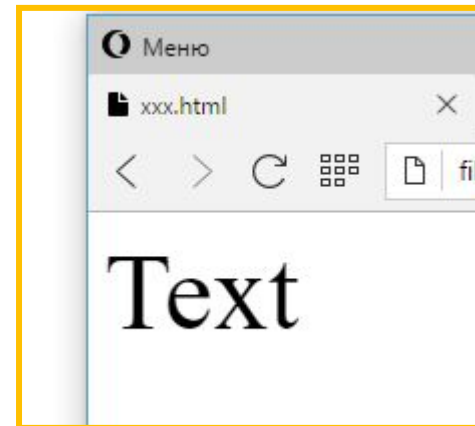
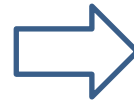
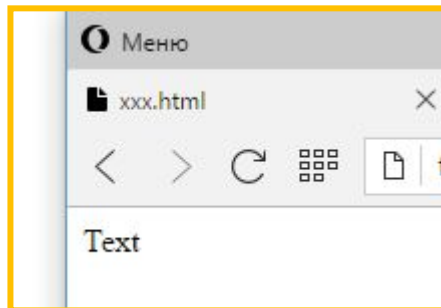
Объекты / Objects



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

```
1 <script>
2   var ob = {
3     name: "Ivan",
4
5     address: {
6       city: "Kiev",
7       street: "Khreshatik",
8       building: 45,
9       show_adr: function() {
10        console.log(this.city, this.street, this.building);
11      }
12    },
13
14    say_hi: function() {
15      console.log("I'm " + this.name + ", my address: " + this.address.city
16        + ", " + this.address.street + ", " + this.address.building);
17    }
18  }
19  ob.say_hi();
20
21  console.log("---");
22
23  ob.address.show_adr();
24 </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 сложный объект и объект «контейнер». Такое разделение называют – цепочкой вызова, длина такой цепочки ограничена только тем сколько объектов участвуют в «матрёшке».

Объекты / Objects

```
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 / Objects

```
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 / Objects

```
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 u null

Объекты / Objects

```
1 <script>
2
3   var a = 7;
4
5   var b = a;
6
7   b++;
8
9   console.log(a, b);
10
11 </script>
```

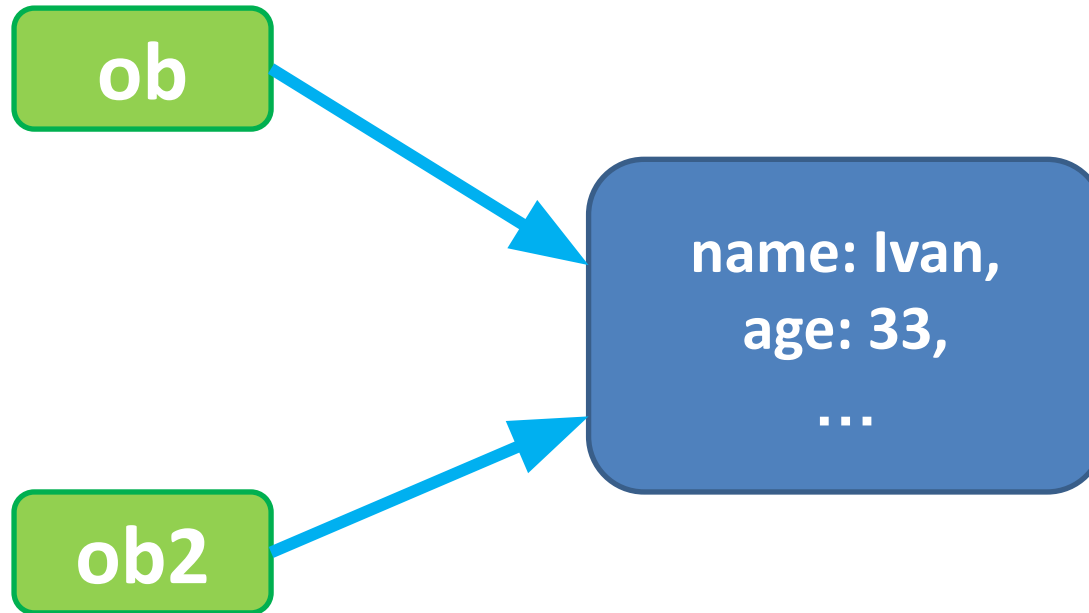
?!?

```
1 <script>
2
3   var ob = {
4     a: 7
5   }
6
7   var ob2 = ob;
8
9   console.log(ob, ob2);
10
11  ob.a++;
12
13  console.log(ob, ob2);
14
15 </script>
```

?!?

object - ссылочная структура данных

Объекты / Objects



object - ссылочная структура данных

Конструктор

```
1 <script>
2   function PersonConstructor () {
3     this.name   = "Ivan";
4     this.age    = 28;
5     this.phone  = "+380505555555";
6     this.say_hello = function () {
7       alert("Person: " + this.name)
8     }
9   }
10
11   var person1 = new PersonConstructor ();
12   var person2 = new PersonConstructor ();
13
14   console.log (person1, person2);
15
16   person1.age = 55;
17
18   console.log (person1, person2);
19 </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

```
1 <script>
2     function PersonConstructor(p_name, p_age, p_phone){
3         this.name    = p_name;
4         this.age     = p_age;
5         this.phone   = p_phone;
6         this.say_hello = function(){
7             console.log("Person: " + this.name, "age: " + this.age, "phone: "
8                 + this.phone);
9         }
10    }
11
12    var person1 = new PersonConstructor("Elena", 22, "0677777775");
13    var person2 = new PersonConstructor("Irina", 30, "0634422354");
14
15    person1.say_hello();
16    person2.say_hello();
17 </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

```
1 <script>
2   var company_info = {
3       company_name:  "Kyivstar",
4       company_adr:   "Kiev, Shevchenko street 45"
5   }
6
7   function PersonConstructor(p_name, p_age, p_phone){
8       this.name     = p_name;
9       this.age      = p_age;
10      this.phone     = p_phone;
11      this.say_hello = function(){
12          console.log("Person: " + this.name, "age: " + this.age, "phone: "
13                      + this.phone, "company: " + this.company_name, "company adr: " +
14                      this.company_adr);
15      }
16      this.__proto__ = company_info;
17  }
18
19  var person1 = new PersonConstructor("Elena", 22, "0677777775");
20  var person2 = new PersonConstructor("Irina", 30, "0634422354");
21
22  person1.say_hello();
23  person2.say_hello();
24
25  company_info.company_name = "McDonalds";
26  company_info.company_adr  = "Kyiv, Ivanova street, 32";
27
28  person1.say_hello();
29  person2.say_hello();
30 </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

```
1 <script>
2   var base_ob = {
3     a: 77
4   }
5
6   var ob = {
7     b: 88,
8     __proto__: base_ob
9   }
10
11  console.log(ob.a, base_ob.a);
12
13  ob.a = 99;
14
15  console.log(ob.a, base_ob.a);
16
17  console.log(ob, base_ob);
18 </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>
```



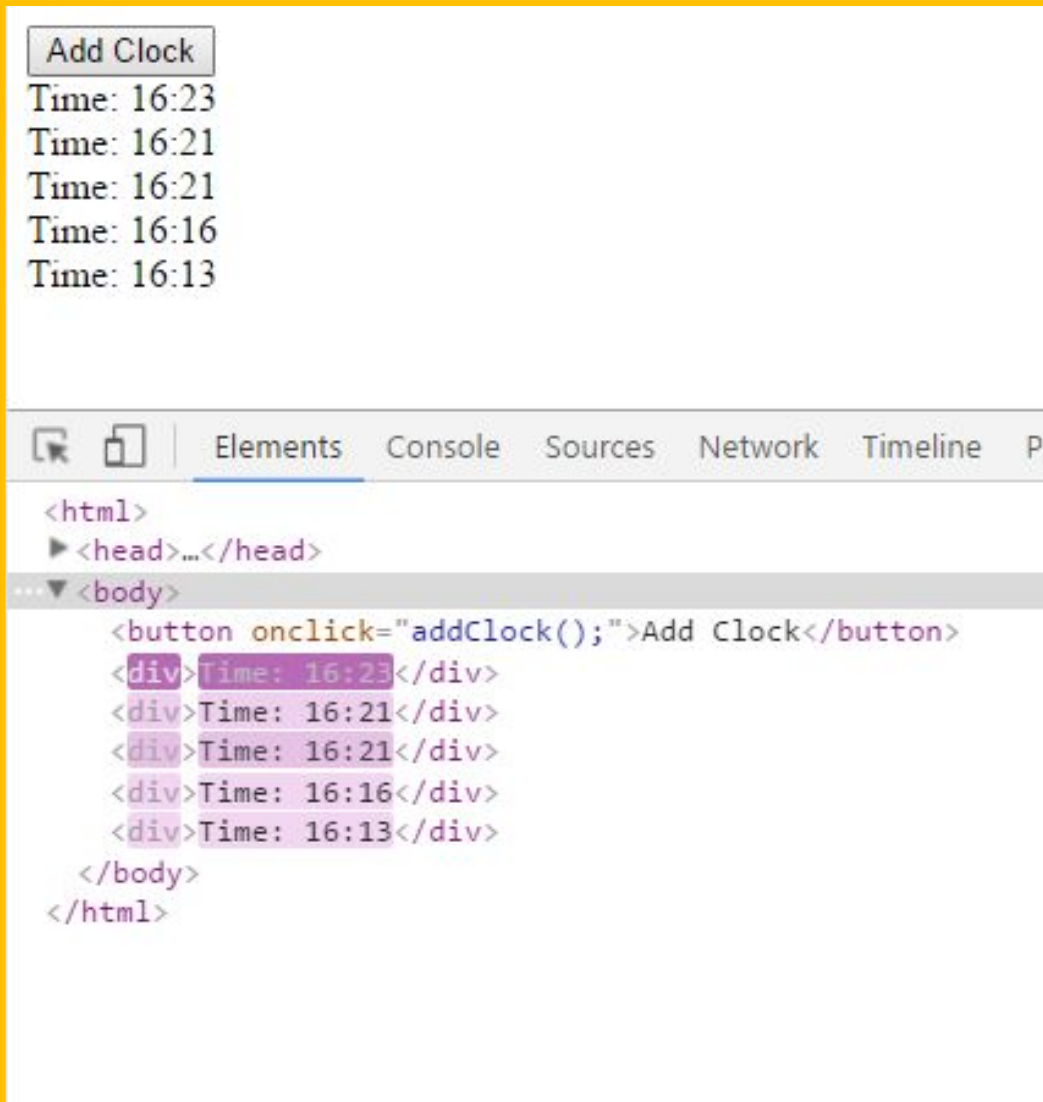
Прототип – объект родитель, дополняет своими свойствами дочерний объект.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script>
5      function ClockConstructor(h, m){
6          this.hours      = h;
7          this.minutes    = m;
8          this.tag        = document.createElement("div");
9          document.body.appendChild(this.tag);
10
11         this.updateShow = function(){
12             this.tag.innerHTML = "Time: " + this.hours + ":"+this.minutes;
13         }
14
15         this.tick = function(){
16             this.minutes++;
17             if(this.minutes > 59) {
18                 this.minutes = 0;
19                 this.hours++;
20             }
21             if(this.hours > 23){
22                 this.hours = 0;
23             }
24             this.updateShow();
25         }
26
27         setInterval(function(thisClock){
28             thisClock.tick();
29         }, 1000, this);
30     }
31
32     function addClock(){
33         var clock = new ClockConstructor(15, 23);
34     }
35 </script>
36 </head>
37 <body>
38     <button onclick="addClock();">Add Clock</button>
39 </body>
40 </html>

```

Объекты «Часы»



The screenshot displays a web browser window with a simple interface. At the top, there is a button labeled "Add Clock". Below the button, five lines of text show the current time: "Time: 16:23", "Time: 16:21", "Time: 16:21", "Time: 16:16", and "Time: 16:13".

Below the browser window, the developer tools are open to the "Elements" tab. The DOM tree shows the following structure:

```
<html>  
  <head>...</head>  
  <body>  
    <button onclick="addClock();">Add Clock</button>  
    <div>Time: 16:23</div>  
    <div>Time: 16:21</div>  
    <div>Time: 16:21</div>  
    <div>Time: 16:16</div>  
    <div>Time: 16:13</div>  
  </body>  
</html>
```

JavaScript как язык программирования

его

концепции

Переменные / Типы /

Операции

Ветвления (условные

операторы)

Циклы / Массивы (структуры

данных)

Функци

Объект

ы

Эрик Фримен Элизабет Робсон

Изучаем программирование на JavaScript



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®