

JS: Object

НАШИ ПРАВИЛА



Включенная камера



Вопросы по поднятой руке



Не перебиваем друг друга



Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору



Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

ЦЕЛЬ

Научиться работать с объектом, узнать соответствующий синтаксис

ПЛАН ЗАНЯТИЯ

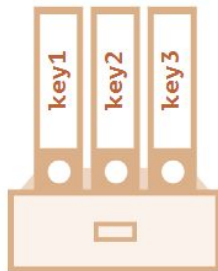
- Создание объекта
- Работа с ключами
- Reference type

Объекты используются для хранения коллекций различных значений и более сложных сущностей.

В JavaScript объекты используются очень часто, это одна из основ языка.

Объект может быть создан с помощью фигурных скобок `{...}` с необязательным списком свойств.

Свойство – это пара «**ключ: значение**», где **ключ** – это строка (также называемая «именем свойства»), а **значение** может быть чем угодно.



Пустой объект можно создать, используя один из двух вариантов синтаксиса:

```
1 let user = new Object(); // синтаксис "конструктор объекта"  
2 let user = {}; // синтаксис "литерал объекта"
```



При использовании литерального синтаксиса `{...}` мы сразу можем поместить в объект несколько свойств в виде пар «ключ: значение»:

```
1 let user = {      // объект
2   name: "John",   // под ключом "name" хранится значение "John"
3   age: 30          // под ключом "age" хранится значение 30
4 };
```

Если в объекте несколько свойств, то они перечисляются через запятую. В объекте `user` сейчас находятся два свойства:

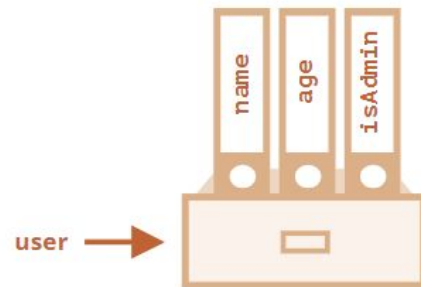
1. Первое свойство с именем `"name"` и значением `"John"`.
2. Второе свойство с именем `"age"` и значением `30`.

Для обращения к свойствам используется запись «через точку»:

```
1 // получаем свойства объекта:  
2 alert( user.name ); // John  
3 alert( user.age ); // 30
```

Значение может быть любого типа. Давайте добавим свойство с логическим значением:

```
1 user.isAdmin = true;
```



Имя свойства может состоять из нескольких слов, но тогда оно должно быть заключено в кавычки:

```
1 let user = {  
2   name: "John",  
3   age: 30,  
4   "likes birds": true // имя свойства из нескольких слов должно быть в кавычках  
5 };
```

Для свойств, имена которых состоят из нескольких слов, доступ к значению «через точку» не работает:

```
1 // это вызовет синтаксическую ошибку  
2 user.likes birds = true
```

Для таких случаев существует альтернативный способ доступа к свойствам через квадратные скобки (**брекет-синтаксис**). Такой способ сработает с любым именем свойства:

```
1  let user = {};  
2  
3  // присваивание значения свойству  
4  user["likes birds"] = true;  
5  
6  // получение значения свойства  
7  alert(user["likes birds"]); // true  
8  
9  // удаление свойства  
10 delete user["likes birds"];
```

Удаление свойств

Удалили из объекта
свойство

```
const person = {  
  name: 'John',  
  age: 25,  
  city: 'Example City'  
};
```

```
delete person.age; // Удаление свойства 'age'
```

Проверка наличия свойств

Проверили наличие
свойства при помощи
оператора **in**

```
const person = {  
  name: 'John',  
  age: 25,  
  city: 'Example City'  
};
```

```
const hasAge = 'age' in person; // true  
const hasGender = 'gender' in person; // false
```

Перебор ключей

Перебор ключей при
помощи цикла

```
const person = {  
  name: 'John',  
  age: 25,  
  city: 'Example City'  
};  
  
// Перебор ключей  
for (let key in person) {  
  console.log(key); // 'name', 'age', 'city'  
}
```

Перебор значений

Перебор значений при помощи цикла.

```
const person = {  
  name: 'John',  
  age: 25,  
  city: 'Example City'  
};  
  
// Перебор значений  
for (let key in person) {  
  console.log(person[key]); // 'John', 25,  
  'Example City'  
}
```

Получение значений/ключей

Получили массив ключей и
массив значений

```
const person = {  
  name: 'John',  
  age: 25,  
  city: 'Example City'  
};  
  
const keys = Object.keys(person); // ['name',  
  'age', 'city']  
const values = Object.values(person); // ['John',  
  25, 'Example City']
```

Reference type

```
let person = {  
  firstName: 'John',  
  lastName: 'Doe'  
};
```



Одно из фундаментальных отличий объектов от примитивов заключается в том, что объекты хранятся и копируются «по ссылке», тогда как примитивные значения: строки, числа, логические значения и т.д. – всегда копируются «как целое значение».

```
1 let message = "Привет!";  
2 let phrase = message;
```

В результате мы имеем две независимые переменные, каждая из которых хранит строку "Привет!".

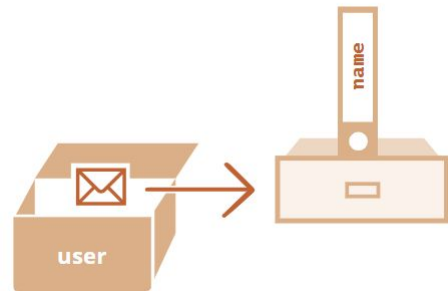


Объекты ведут себя иначе, чем примитивные типы.

Переменная, которой присвоен объект, хранит не сам объект, а его «адрес в памяти» – другими словами, «ссылку» на него.

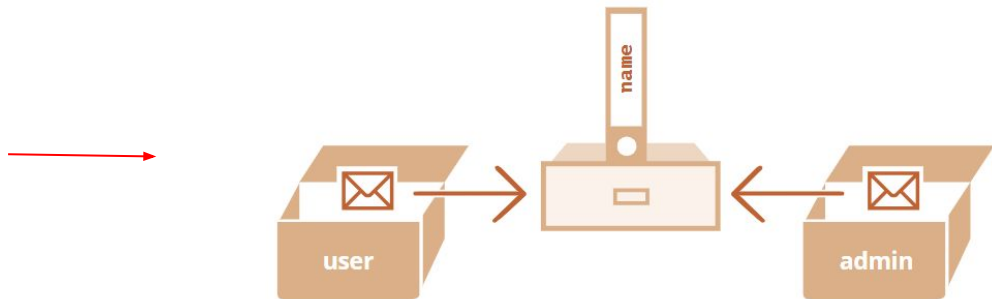
```
1 let user = {  
2   name: "John"  
3 };
```


Вот как это на самом деле хранится в памяти:



При копировании переменной объекта копируется ссылка, но сам объект не дублируется.

```
1 let user = { name: "John" };  
2  
3 let admin = user; // копируется ссылка
```





Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH