**Копирование объектов и ссылки**

строки, числа, логические значения и т.д. – всегда копируются «как целое значение».

let message = "Привет!";

let phrase = message;

результате мы имеем две независимые переменные, каждая из которых хранит строку

 объекты хранятся и копируются «по ссылке»

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

Когда мы выполняем действия с объектом, к примеру, берём свойство user.name, движок JavaScript просматривает то, что находится по этому адресу, и выполняет операцию с самим объектом.

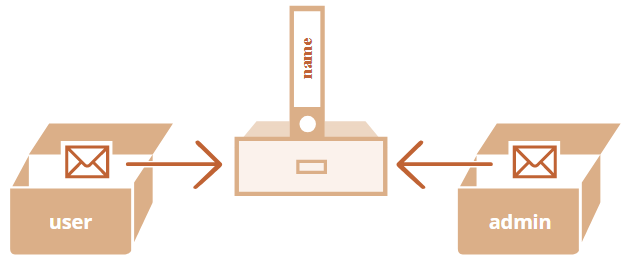
**При копировании переменной объекта копируется ссылка, но сам объект не дублируется.**

Например:

let user = { name: "John" };

let admin = user; // копируется ссылка

Теперь у нас есть две переменные, каждая из которых содержит ссылку на один и тот же объект:



Как вы можете видеть, все ещё есть один объект, но теперь с двумя переменными, которые ссылаются на него.

Теперь мы можем использовать любую переменную для доступа к объекту и изменения его содержимого:

let user = { name: 'John' };

let admin = user;

admin.name = 'Pete'; // изменено по ссылке из переменной "admin"

alert(user.name); // 'Pete', изменения видны по ссылке из переменной "user"

## [Сравнение по ссылке](https://learn.javascript.ru/object-copy" \l "sravnenie-po-ssylke)

Два объекта равны только в том случае, если это один и тот же объект.

Например, здесь a и b ссылаются на один и тот же объект, поэтому они равны:

let a = {};

let b = a; // копирование по ссылке

alert( a == b ); // true, обе переменные ссылаются на один и тот же объект

alert( a === b ); // true

И здесь два независимых объекта не равны, даже если они выглядят одинаково (оба пусты):

let a = {};

let b = {}; // два независимых объекта

alert( a == b ); // false

Для сравнений типа obj1 > obj2 или для сравнения с примитивом obj == 5 объекты преобразуются в примитивы.

## [Клонирование и объединение, Object.assign](https://learn.javascript.ru/object-copy" \l "cloning-and-merging-object-assign)

 в JavaScript нет встроенного метода для клонированя объекта.

Чтобы дублировать объект можно перебрать его свойства при помощи for…in и примитивно скопировать их в новый объект.

let user = {

name: "John",

age: 30

};

let clone = {}; // новый пустой объект

for (let key in user) {

clone[key] = user[key];

}

// теперь clone это полностью независимый объект с тем же содержимым

clone.name = "Pete"; // изменим в нём данные

alert( user.name ); // все ещё John в первоначальном объекте

…..или использовать для этого метод [Object.assign](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Object/assign)

Синтаксис:

Object.assign(dest, [src1, src2, src3...])

* Первый аргумент dest — целевой объект.
* Остальные аргументы src1, ..., srcN (может быть столько, сколько необходимо) являются исходными объектами
* Метод копирует свойства всех исходных объектов src1, ..., srcN в целевой объект dest. Другими словами, свойства всех аргументов, начиная со второго, копируются в первый объект.
* Возвращает объект dest.

Например, мы можем использовать его для объединения нескольких объектов в один:

let user = { name: "John" };

let permissions1 = { canView: true };

let permissions2 = { canEdit: true };

// копируем все свойства из permissions1 и permissions2 в user

Object.assign(user, permissions1, permissions2);

// теперь user = { name: "John", canView: true, canEdit: true }

Если скопированное имя свойства уже существует, оно будет перезаписано:

let user = { name: "John" };

Object.assign(user, { name: "Pete" });

alert(user.name); // теперь user = { name: "Pete" }

Мы также можем использовать Object.assign для замены цикла for..in для простого клонирования:

let user = {

name: "John",

age: 30

};

let clone = Object.assign({}, user);

Он копирует все свойства user в пустой объект и возвращает его.

## [Вложенное клонирование](https://learn.javascript.ru/object-copy" \l "vlozhennoe-klonirovanie)

Как скопировать объект который содердит в себе еще один объект?

Например

let user = {

name: "John",

sizes: {

height: 182,

width: 50

}

};

alert( user.sizes.height ); // 182

Теперь недостаточно просто скопировать clone.sizes = user.sizes, потому что user.sizes – это объект, он будет скопирован по ссылке. Таким образом, clone и user будут иметь общий объект sizes:

let user = {

name: "John",

sizes: {

height: 182,

width: 50

}

};

let clone = Object.assign({}, user);

alert( user.sizes === clone.sizes ); // true, тот же объект

// user и clone обладают общим свойством sizes

user.sizes.width++; // изменяем свойства в первом объекте

alert(clone.sizes.width); // 51, видим результат в другом

Чтобы исправить это, мы должны использовать цикл клонирования, который проверяет каждое значение user[key] и, если это объект, тогда также копирует его структуру. Это называется «глубоким клонированием».

Мы можем реализовать глубокое клонирование, используя рекурсию. Или, чтобы не изобретать велосипед заново, возьмите готовую реализацию, например [\_.cloneDeep(obj)](https://lodash.com/docs#cloneDeep) из библиотеки JavaScript [lodash](https://lodash.com/).

Также мы можем использовать глобальный метод [structuredClone()](https://developer.mozilla.org/en-US/docs/Web/API/structuredClone), который позволяет сделать полную копию объекта. К сожалению он поддерживается только современными браузерами. [Здесь](https://caniuse.com/?search=structuredClone) можно ознакомиться с поддержкой этого метода.