Частное учреждение образования

«Колледж бизнеса и права»

|  |
| --- |
| УТВЕРЖДАЮ  Заведующий  методическим кабинетом  \_\_\_\_\_\_\_\_\_\_Е.В.Паскал  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ |

|  |  |
| --- | --- |
| Специальность:  2-40 01 01 «Программное обеспечение информационных технологий» | Дисциплина: «Скриптовые языки программирования» |

**Лабораторная работа № 11**

Инструкционно-технологическая карта

Тема: Копирование по ссылке и по значению.

Цель: Научиться копированию по ссылке и по значению.

Время выполнения: 2 часа

# Порядок выполнения работы

1. Изучить теоретические сведения к лабораторной работе.

2. Реализовать алгоритм решения задачи.

3. Отлаженную, работающую программу сдать преподавателю. Работу программы показать с помощью самостоятельно разработанных тестов.

4. Ответить на контрольные вопросы.

# Теоретические сведения

## Копирование объектов и ссылки

Одним из фундаментальных отличий объектов от примитивных типов данных является то, что они хранятся и копируются «по ссылке».

Примитивные типы: строки, числа, логические значения – присваиваются и копируются «по значению».

Например:

let message = "Привет!";

let phrase = message;

В результате будет две независимые переменные, каждая из которых хранит строку «Привет!».

Объекты ведут себя иначе.

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

let user = {

  name: "Иван"

};

Сам объект хранится где-то в памяти. А в переменной user лежит «ссылка» на эту область памяти.

Когда переменная объекта копируется – копируется ссылка, сам же объект не дублируется.

Если представить объект как ящик, то переменная – это ключ к нему. Копирование переменной дублирует ключ, но не сам ящик.

Например:

let user = { name: "Иван" };

let admin = user; // копируется ссылка

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

let user = { name: 'Иван' };

let admin = user;

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

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

Приведённый выше пример демонстрирует, что объект только один. Как если бы был один ящик с двумя ключами и используем один из них (admin), чтобы войти в него и что-то изменить, а затем, открыв ящик другим ключом (user), можно было бы увидеть эти изменения.

## Сравнение по ссылке

Операторы равенства == и строгого равенства === для объектов работают одинаково.

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

В примере ниже две переменные ссылаются на один и тот же объект, поэтому они равны друг другу:

let a = {};

let b = a; // копирование по ссылке

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

alert( a === b ); // true

В другом примере два разных объекта не равны, хотя 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 объектов, Object.assign

Таким образом, при копировании переменной с объектом создаётся ещё одна ссылка на тот же самый объект.

Но что, если нам всё же нужно дублировать объект? Создать независимую копию, клон?

Это выполнимо, но немного сложно, так как в JavaScript нет встроенного метода для этого. На самом деле, такая нужда возникает редко. В большинстве случаев нам достаточно копирования по ссылке.

Но если действительно этого хочется, то нужно создавать новый объект и повторять структуру дублируемого объекта, перебирая его свойства и копируя их.

Например так:

let user = {

  name: "Иван",

  age: 30

};

let clone = {}; // новый пустой объект

// скопируем все свойства user в него

for (let key in user) {

  clone[key] = user[key];

}

// теперь в переменной clone находится абсолютно независимый клон объекта

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

alert( user.name ); // в оригинальном объекте значение свойства `name` осталось прежним – Иван.

Кроме того, для этих целей можно использовать метод Object.assign.

Синтаксис:

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

1. Первый аргумент dest ‑ целевой объект.
2. Остальные аргументы src1, ..., srcN (может быть столько, сколько нужно) являются исходными объектами
3. Метод копирует свойства всех исходных объектов src1, ..., srcN в целевой объект dest. То есть, свойства всех перечисленных объектов, начиная со второго, копируются в первый объект.
4. Возвращает объект dest.

let user = { name: "Иван" };

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: "Иван", canView: true, canEdit: true }

Если принимающий объект (user) уже имеет свойство с таким именем, оно будет перезаписано:

let user = { name: "Иван" };

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

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

Также можно использовать Object.assign для замены for..in на простое клонирование:

let user = {

  name: "Иван",

  age: 30

};

let clone = Object.assign({}, user);

Этот метод скопирует все свойства объекта user в пустой объект и возвратит его.

## Вложенное клонирование

Может показаться, что все свойства объекта user хранят примитивные значения. Но свойства могут быть ссылками на другие объекты. Что с ними делать?

Например, есть объект:

let user = {

  name: "Иван",

  sizes: {

    height: 182,

    width: 50

  }

};

alert( user.sizes.height ); // 182

Теперь при клонировании недостаточно просто скопировать clone.sizes = user.sizes, поскольку user.sizes – это объект, он будет скопирован по ссылке. А значит объекты clone и user в своих свойствах sizes будут ссылаться на один и тот же объект:

let user = {

  name: "Иван",

  sizes: {

    height: 182,

    width: 50

  }

};

let clone = Object.assign({}, user);

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

// user и clone обращаются к одному sizes

user.sizes.width++;       // меняем свойство в одном объекте

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

Чтобы исправить это, надо в цикле клонирования делать проверку, не является ли значение user[key] объектом, и если это так – скопировать и его структуру тоже. Это называется «глубокое клонирование».

Можно реализовать глубокое клонирование, используя рекурсию. Или, чтобы не изобретать велосипед, использовать готовую реализацию – метод \_.cloneDeep(obj) из JavaScript-библиотеки lodash.

## Итого

Объекты присваиваются и копируются по ссылке. Другими словами, переменная хранит не «значение объекта», а «ссылку» (адрес в памяти) на это значение. Поэтому копирование такой переменной или передача её в качестве аргумента функции приводит к копированию этой ссылки, а не самого объекта.

Все операции с использованием скопированных ссылок (например, добавление или удаление свойств) выполняются с одним и тем же объектом.

Для «простого клонирования» объекта можно использовать Object.assign. Необходимо помнить, что Object.assign не делает глубокое клонирование объекта. Если внутри копируемого объекта есть свойство, значение которого не является примитивом, оно будет передано по ссылке. Для создания «настоящей копии» (полного клона объекта) можно воспользоваться методом из сторонней JavaScript-библиотеки \_.cloneDeep(obj).

# Контрольные вопросы

1. Чем отличается копирование примитивных типов от копирования объектов?
2. Привести пример копирования объекта.
3. Что такое вложенное клонирование и для чего оно применяется?

# Задания

**Задание 1. Создать** объект с данным набором полей и **скопировать** его по **ссылке** и по **значению** в новый объект. Вывести содержимое трёх объектов.

| **№** | **Задание** |
| --- | --- |
|  | Описать объект с именем GROUP, содержащий поля: Name – фамилия и инициалы, DAT – дата рождения (год, месяц, число), SES – успеваемость (массив из трех элементов). |
|  | Описать объект с именем STUDENT, содержащий поля: Name – фамилия и инициалы, Kurs − курс, SES – успеваемость (массив из пяти элементов). |
|  | Описать объект с именем STUD, содержащий поля: Name – фамилия и инициалы, GROUP – название группы (факультет, курс, номер группы), SES – успеваемость (массив из четырёх элементов). |
|  | Описать объект с именем NOTE, содержащий поля: Name – фамилия и инициалы, TELE – номер телефона, DATE – дата рождения (год, месяц, число). |
|  | Описать объект с именем NOTE1, содержащий поля: Name – фамилия и инициалы, TELE – номер телефона, DATE – дата рождения (год, месяц, число). |
|  | Описать объект с именем NOTE2, содержащий поля: Name – фамилия и инициалы, TELE – номер телефона, DATE – дата рождения (год, месяц, число). |
|  | Описать объект с именем PERSON, содержащий поля: Name – фамилия и инициалы, FAC – факультет, GROUP – группа, DATE – дата поступления в ВУЗ (год, месяц, число). |
|  | Описать объект с именем ZNAK, содержащий поля: Name – фамилия и имя, ZOD – знак зодиака, DATE – дата рождения (массив из трех чисел: год, месяц, число). |
|  | Описать объект с именем STUDENT, содержащий поля:  фамилия и инициалы студента;  номер группы;  успеваемость (массив из четырех оценок на экзаменах в 5-бальной системе). |
|  | Описать объект с именем TOVAR, содержащий поля:  название товара;  количество единиц товара;  стоимость товара;  дата  поступления товара в виде структуры (год, месяц, день). |
|  | Описать объект с именем MARSHRUT, содержащий поля:  номер маршрута;  начальный пункт маршрута;  конечный пункт  маршрута;  длина маршрута. |
|  | Описать объект с именем ABON, содержащий поля:  фамилия и инициалы абонента;  номер телефона;  дата подключения телефона в виде структуры (год, месяц, день);  начисленная сумма оплаты;  сумма на счету абонента. |

**Литература**

**Диков, А.В.** Клиентские технологии веб программирования: JavaScript и DOM: учебное пособие / А.В. Диков. – СПб: Лань, 2020 – 124 с.

**Читанамбри, Кирупа.** Изучаем React / Кирупа Читанамбри – 2-е изд. – М.: Эксмо, 2019. 368 с.

**Хавербеке, Марейн.** Выразительный JavaScript. Современное веб-программирование. / Марейн Хавербеке – 3-е изд. – СПб.: Питер, 2019. – 480 с.:

Преподаватель Рогалевич А.В.

Рассмотрено на заседании цикловой

комиссии программного обеспечения

информационных технологий №10

Протокол № \_\_\_ от \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Председатель ЦК\_\_\_\_\_\_\_\_ В.Ю.Михалевич