# Лабораторная работа № 6 Работа с объектами

**Цель работы:** получить навыки создания объектов, доступа к полям и методам, ознакомится с прототипным наследованием.

**Краткие теоретические сведения:**

На сегодняшний день объектно-ориентированное программирование (ООП) на сегодняшний день является одной из господствующих парадигм в разработке приложений. JavaScript предоставляет возможности ООП, но имеет некоторые особенности.

Основным понятием ООП является объект – сложная комплексная структура, объединяющая в себе данные и методы их обработки.

Для работы с подобными структурами в JavaScript используются объекты. Каждый объект может хранить свойства, которые описывают его состояние, и методы, которые описывают его поведение.

**Создание нового объекта**

Есть несколько способов создания нового объекта.

Первый способ заключается в использовании конструктора Object:

const user = new Object();

В данном случае объект называется user. Он определяется также, как и любая обычная переменная. Выражение new Object() представляет вызов конструктора – функции, создающей новый объект. Для вызова конструктора применяется оператор new. Вызов конструктора фактически напоминает вызов обычной функции.

Второй способ создания объекта представляет использование фигурных скобок – объектного литерала:

const user = {};

На сегодняшний день более распространенным является второй способ.

**Свойства и методы объекта**

После создания объекта мы можем определить в нем свойства. Чтобы определить свойство, надо после названия объекта через точку указать имя свойства и присвоить ему значение, или использовать альтернативный способ с помощью синтаксиса массивов:

const user = {};

user.name = "Alex";

user.age = 21;

user["group"] = "IP-31"

В данном случае объявляются три свойства name, age и group, которым присваиваются соответствующие значения. После этого мы можем использовать эти свойства, например, вывести их значения в консоли:

Методы объекта определяют его поведение или действия, которые он производит. Методы представляют собой функции. Например, определим метод, который бы выводил имя и возраст человека:

user.info = function(){

console.log(user.name);

console.log(user.age);

};

Также свойства и методы могут определяться непосредственно при определении объекта:

var user = {

name: "Alex",

age: 21,

group: "IP-31",

info: function(){

console.log(this.name);

console.log(this.age);

}

};

Чтобы обратиться к свойствам или методам объекта внутри этого объекта, используется ключевое слово this. Оно означает ссылку на текущий объект.

**Конструктор объектов**

Кроме создания новых объектов JavaScript предоставляет нам возможность создавать новые типы объектов с помощью конструкторов.

Конструктор позволяет определить новый тип объекта. Тип представляет собой абстрактное описание или шаблон объекта. Определение типа может состоять из функции конструктора, методов и свойств.

Для начала определим конструктор:

function User(pName, pAge) {

this.name = pName;

this.age = pAge;

this.info = function(){

document.write("Имя: " + this.name + "; возраст: " + this.age + "<br/>");

};

}

Конструктор – это обычная функция за тем исключением, что в ней мы можем установить свойства и методы. Для установки свойств и методов используется ключевое слово this. В данном случае устанавливаются два свойства name и age и один метод info. Как правило, названия конструкторы в отличие от названий обычных функций начинаются с большой буквы.

После этого в программе мы можем определить объект типа User и использовать его свойства и методы:

const alex = new User("Alex", 21);

console.log(alex.name); // Alex

alex.info();

Оператор **instanceof** позволяет проверить, с помощью какого конструктора создан объект. Если объект создан с помощью определенного конструктора, то оператор возвращает true:

const alex = new User("Alex", 21);

console.log(alex instanceof User); // true

**Расширение объектов. Prototype**

Кроме непосредственного определения свойств и методов в конструкторе мы также можем использовать свойство prototype. Каждая функция имеет свойство prototype, представляющее прототип функции. То есть свойство User.prototype представляет прототип объектов User. И любые свойства и методы, которые будут определены в User.prototype, будут общими для всех объектов User.

function User(pName, pAge) {

this.name = pName;

this.age = pAge;

this.info = function() {

document.write(

"Имя: " + this.name + "; возраст: " + this.age + "<br/>"

);

};

}

User.prototype.hello = function() {

document.write(this.name + " говорит: 'Привет!'<br/>");

};

User.prototype.group = "IP-31";

const alex = new User("Alex", 21);

alex.hello();

const max = new User("Max", 21);

max.hello();

console.log(max.group);

**Классы**

С внедреием стандарта ES2015 (ES6) в JavaScript появился новый способ определения объектов – с помощью классов. Класс представляет описание объекта, его состояния и поведения, а объект является конкретным воплощением или экземпляром класса.

Для определения класса используется ключевое слово class.

После этого мы можем создать объекты класса с помощью конструктора:

class Person{}

const alex = new Person();

const max = new Person();

По умолчанию классы имеют один конструктор без параметров. Поэтому в данном случае при вызове конструктора в него не передается никаких аргументов. Однако есть возможность определить в классе свои конструкторы. Также класс может содержать свойства и методы:

class Person{

constructor(name, age){

this.name = name;

this.age = age;

}

info(){

console.log(this.name, this.age);

}

}

const tom = new Person("Alex", 21);

alex.info(); // Tom 34

console.log(alex.name); // Tom

Конструктор определяется с помощью метода с именем constructor. По сути это обычный метод, который может принимать параметры. Основная цель конструктора – инициализировать объект начальными данными. И в данном случае в конструктор передаются два значения – для имени и возраста пользователя.

Для хранения состояния в классе определяются свойства. Для их определения используется ключевое слово this. В данном случае в классе два свойства: name и age.

Поведение класса определяют методы. В данном случае определен метод info(), который выводит значения свойств на консоль.

**Наследование**

Одни классы могут наследоваться от других. Наследование позволяет сократить объем кода в классах-наследниках. Например, определим следеющие классы:

class Person{

constructor(name, age){

this.name = name;

this.age = age;

}

info(){

console.log(this.name, this.age);

}

}

class Employee extends Person{

constructor(name, age, company){

super(name, age);

this.company = company;

}

info(){

super.info();

console.log("Employee in", this.company);

}

work(){

console.log(this.name, "is hard working");

}

}

const bob = new Person("Bob", 34);

const bill = new Employee("Bill", 64, "Microsoft");

bob.info();

bill.info();

bill.work();

Для наследования одного класса от другого в определении класса применяется оператор extends, после которого идет название базового класса. То есть в данном случае класс Employee наследуется от класса Person. Класс Person еще называется базовым классом, классом-родителем, суперклассом, а класс Employee – классом-наследником, подклассом, производным классом.

Производный класс, как и базовый, может определять конструкторы, свойства, методы. Вместе с тем с помощью слова super производный класс может ссылаться на функционал, определенный в базовом. Например, в конструкторе Employee можно вручную не устанавливать свойства name и age, а с помощью выражения super(name, age); вызвать конструктор базового класса и тем самым передать работу по установке этих свойств базовому классу.

**Статические методы**

Статические методы вызываются для всего класса вцелом, а не для отдельного объекта. Для их определения применяется оператор static. Например:

class Person{

constructor(name, age){

this.name = name;

this.age = age;

}

static nameToUpper(person){

return person.name.toUpperCase();

}

info(){

console.log(this.name, this.age);

}

}

const tom = new Person("Tom Soyer", 34);

let personName = Person.nameToUpper(tom);

console.log(personName); // TOM SOYER

В данном случае определен статический метод nameToUpper(). В качестве параметра он принимает объект Person и переводит его имя в верхний регистр. Поскольку статический метод относится классу вцелом, а не к объекту, то мы НЕ можем использовать в нем ключевое слово this и через него обращаться к свойствам объекта.

**Практическая часть:**

**Задание 1**. В соответствии со своим вариантом определить объект, с заданными полями и методом вывода информации об объекте. Объект создается с помощью объектного литерала.

| Вариант | Описание | Свойство |
| --- | --- | --- |
| 01 | Процессор | Тактовая частота  Количество ядер  Разрядность процессора  Техпроцесс |
| 02 | Память | Тип памяти  Объем  Скорость чтения  Скорость записи |
| 03 | Корпус | Мощность блока питания  Тип корпуса  Форм-фактор материнской платы  Охлаждение |
| 04 | Жесткий диск | Тип накопителя  Объем  Интерфейс  Скорость вращения шпинделя |
| 05 | SSD диск | Объем  Интерфейс  Тип микросхем Flash  Контроллер |
| 06 | Оптический носитель | Тип установки  Тип  Интерфейс подключения  Максимальная скорость чтения  Максимальная скорость чтения |
| 07 | Клавиатура | Тип  Технология переключения  Назначение  Интерфейс подключения |
| 08 | Мышь | Интерфейс подключения  Назначение  Тип сенсора  Модель сенсора  Максимальное разрешение сенсора |
| 09 | Звуковая карта | Аналоговые выходы  Аналоговые входы  Микрофонные входы  Тип  Количество каналов  Интерфейс подключения |
| 10 | Видеокарта | Интерфес  Производитель графического процессора  Графический процессор  Частота графического процессора  Видеопамять (объем) |
| 11 | Блок бесперебойного питания | Тип  AVR  Мощность  Время автономного питания |
| 12 | Планшет | Диагональ экрана  Процессор  Разрешение экрана  Операционная система  Оперативная память  Внутренняя память |
| 13 | Электронная книга | Размер экрана  Тип экрана  Разрешение экрана  Флэш-память |
| 14 | Акустика | Тип  Усилитель  Максимальная мощность  Материал корпуса |
| 15 | Фитнес-браслет | Объем оперативной памяти  Поддерживаемые платформы  Датчики |
| 16 | Аккумулятор | Тип аккумулятора  Емкость  Напряжение  Полярность |
| 17 | Мобильный телефон | Тип  Операционная система  Размер экрана  Разрешение экрана  Оперативная память |
| 18 | Радиоприемник | Питание  Тип  Выходная мощность звука  Тип тюнера  Модуляция |
| 19 | Телевизор | Тип  Диагональ  Частота  Разрешение |
| 20 | Фотоаппарат | Тип камеры  Тип матрицы  Размер экрана  Количество точекматрицы  Максимальное разрешение видео |
| 21 | Видеокамера | Разрешение видеосъемки  Формат сжатия видео  Размер экрана  Количество точек матрицы |
| 22 | Принтер | Тип  Формат  Количество цветов  Технология печати  Скорость печати |
| 23 | Игровая приставка | Тип  Объем накопителя  Максимальное разрешение в играх  Оптический привод |
| 24 | IP камера | Тип матрицы  Конструкция  Стандарты беспроводной связи  Сетевой интерфейс |
| 25 | Портативная радиостанция | Стандарт  Количество каналов  Дальность радиосвязи  Количество радиостанций в комплекте |
| 26 | Модем | Тип  Интерфейс подключения  Поддержка сотовой связи  Поддержка карт памяти |
| 27 | Монитор | Диагональ  Соотношение сторон  Разрешение  Частота обновления экрана |
| 28 | Велосипед | Класс  Материал рамы  Тип трансмиссии  Диаметр колес |
| 29 | Микроволновая печь | Тип  Объем  Потребляемая мощность  Выходная мощность микроволн |
| 30 | Стиральная машина | Загрузка белья  Максимальная загрузка  Класс энергопотребления  Класс стирки  Максимальная скорость отжима |

**Задание 2.** Написать функцию-конструктор для создания объектов в соответствии с вариантом. Определить геттеры и сеттеры для получения доступа к свойствам. В прототип объектов добавить свойство – дата выхода на рынок, метод отображения информации об объекте.

**Задание 3.** Описать класс, описывающий объекты в соответствии с вариантом. Предусмотреть конструктор с параметрами, геттеры и сеттеры для получения доступа к свойствам, метод отображения информации об объекте.

**Задание 4.** Описать класс, являющийся наследником класса задания 3. В производном классе добавить свойства: дата выхода на рынок, стоимость. Переопределить метод вывода информации об объекте.

**Задание 5.** На основе разработанного класса задания 4 создать массив объектов. Определить объекты с максимальной и минимальной стоимостью, суммарную стоимость всех объектов, вычислить среднюю стоимость объекта, подсчитать количество объектов со стоимостью выше средней. Для выполнения задания использовать встроенные методы массивов и объект Math.

**Требования по содержанию отчета:**

Отчет должен включать следующие пункты:

1. Титульный лист.
2. Цель работы.
3. Задание.
4. Листинг программного кода с комментариями.
5. Результаты выполнения скрипта.
6. Выводы.

**Контрольные вопросы для защиты работ:**

1. Как создать объект в JavaScript?
2. Что такое функция конструктор?
3. Что такое Prototype?
4. Как создать метод объекта?
5. Как преребрать свойства объекта?
6. Как определить собственные свойства объекта?
7. Как добавить, удалить свойство?
8. Как определить класс?
9. Как определить класс, наследуемый от заданного?
10. Как обратиться к функционалу базового класса из производного?