Лабораторна робота №6. Робота з об'єктами

Мета роботи: отримати навички створення об'єктів, доступу до полів та методів, ознайомитись із прототипним успадкуванням.

Теоретична частина

На сьогоднішній день об'єктно-орієнтоване програмування (ООП) є однією з масових парадигм у розробці додатків. JavaScript надає можливості ОВП, але має деякі особливості. Основним поняттям ООП є об'єкт – складна комплексна структура, що поєднує дані та методи їх обробки.

Для роботи з подібними структурами JavaScript використовуються об'єкти. Кожен об'єкт може зберігати властивості, що описують його стан, та методи, що описують його поведінку.

Створення нового об'єкту

Існує кілька способів створення нового об'єкта.

Перший спосіб полягає у використанні конструктора `Object` :

```
const user = new Object();
```

В даному випадку об'єкт називається user. Він визначається також, як і будь-яка звичайна змінна. Вираз `new Object()` представляє виклик конструктора – функції, що створює новий об'єкт. Для виклику конструктора використовується оператор `new`. Виклик конструктора фактично нагадує виклик звичайної функції.

Другий спосіб створення об'єкта представляє використання фігурних дужок – об'єктного літералу:

```
const user = {};
```

На сьогоднішній день найпоширенішим є другий спосіб.

Властивості та методи об'єкта

Після створення об'єкта можна визначити властивості. Щоб визначити властивість, треба після назви об'єкта через точку вказати ім'я властивості та привласнити йому значення, або використовувати альтернативний спосіб за допомогою синтаксису масивів:

```
const user = {};
user.name = "Alex";
user.age = 21;
user ["group"] = "KI-41"
    console.log(user);
```

У цьому прикладі оголошуються три властивості `name`, `age` і `group`, яким привласнюються відповідні значення. Після цього можна використовувати ці властивості, наприклад вивести їх значення в консолі.

Методи об'єкта визначають його поведінку чи дії, що він виробляє. Методи представляють собою функції. Наприклад, визначимо метод, який би виводив ім'я та вік людини:

```
const user = {};
user.name = "Alex";
user.age = 21;
user ["group"] = "KI-41"
var info = function() {
    console.log(user.name);
    console.log(user.age);
};
info(user);
```

Також властивості та методи можуть визначатися безпосередньо при визначенні об'єкта:

```
var user = {
    name: "Alex",
    age: 21,
    group: "KI-41",
    info: function(){
        console.log(this.name);
        console.log(this.age);
    }
};
console.log(user);
```

Щоб звернутися до властивостей або методів об'єкта всередині цього об'єкта, використовується ключове слово `this`. Це означає посилання на поточний об'єкт.

Конструктор об'єктів

Крім створення нових об'єктів JavaScript надає можливість створювати нові типи об'єктів за допомогою конструкторів. Конструктор дає змогу визначити новий тип об'єкта. Тип є абстрактним описом або шаблоном об'єкта. Визначення типу може складатися з функції конструктора, методів та властивостей.

Для початку визначимо конструктор:

```
function User (pName, pAge) {
      this.name=pName;
      this.age=pAge;
      this.info = function(){
            document.write ("Ім'я: " + this.name + "; вік: " + this.age +
      " " );
      };
}
```

Конструктор - це проста функція за тим винятком, що в ній ми можемо встановити властивості та методи. Для встановлення властивостей та методів використовується ключове слово this. В даному випадку встановлюються дві властивості name і age та один метод info . Як правило, назви конструктори, на відміну від назв звичайних функцій, починаються з великої літери.

Після цього у програмі можна визначити об'єкт типу User і використовувати його властивості та методи:

```
const alex = new User ("Alex", 21);
console.log(alex.name); // Alex alex.info();
```

Оператор `instanceof` дозволяє перевірити, за допомогою якого конструктора створено об'єкт. Якщо об'єкт створено за допомогою певного конструктора, оператор повертає `true`:

```
const alex = new User ("Alex", 21);
console.log(alex instanceof User); // true
```

Розширення об'єктів. Prototype

Крім безпосереднього визначення властивостей та методів у конструкторі ми також можемо використовувати властивість `prototype`. Кожна функція має властивість `prototype`, що представляє прототип функції. Тобто властивість `User.prototype` представляє прототип об'єктів User. І будь-які властивості та методи, які будуть визначені в `User.prototype`, будуть спільними для всіх об'єктів User .

```
function User (pName, pAge) {
      this.name=pName;
      this.age=pAge;
      this.info=function() {
            document.write("Ім'я: " + this.name + "; вік: " + this.age +
"<br />");
      };
}
User.prototype.hello = function () {
      document.write (this.name + " каже: 'Привіт!'< 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);
```

Класи

З впровадженням стандарту ES2015 (ES6) в JavaScript з'явився новий спосіб визначення об'єктів - за допомогою класів. Клас представляє опис об'єкта, його стану та поведінки, а об'єкт є конкретним втіленням або екземпляром класу. Для визначення класу використовується ключове слово `class`.

Після цього можна створити об'єкти класу за допомогою конструктора:

```
class Person {}
const alex = new Person();
const max = new Person();
```

За замовчуванням, класи мають один конструктор без параметрів. Тому в даному випадку при виклику конструктора до нього не передається жодних аргументів. Проте є можливість визначити у класі своїх конструкторів. Також клас може містити властивості та методи:

```
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 визначається за допомогою методу `constructor`. По суті, це звичайний метод, який може приймати параметри. Основна мета конструктора – ініціалізувати об'єкт початковими даними. І в даному випадку в конструктор передаються два значення – для імені та віку користувача.

Для зберігання стану у класі визначаються властивості. Для визначення використовується ключове слово this . У разі у класі дві властивості: name і age .

Поведінка класу визначають методи. В даному випадку визначено метод info ( ), який виводить значення властивостей консоль.

Успадкування

Одні класи можуть успадковуватись від інших. Спадкування дозволяє скоротити обсяг коду в класах-спадкоємцях. Наприклад, визначимо такі класи:

```
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 успадкування одного класу від іншого визначення класу застосовується оператор `extends`, після якого йде назва базового класу. Тобто в даному випадку клас Employee успадковується від класу Person . Клас Person ще називається базовим класом, класом-батьком, суперкласом, а клас Employee – класом-спадкоємцем, підкласом, похідним класом.

Похідний клас, як і базовий, може визначати конструктори, властивості, методи. Разом з тим, за допомогою слова super похідний клас може посилатися на функціонал, визначений у базовому. Наприклад, в конструкторі Employee можна вручну не встановлювати властивості name і age , а за допомогою виразу super (name , age); викликати конструктор базового класу і цим передати роботу з встановлення цих властивостей базовому класу.

Статичні методи

Статичні методи викликаються для всього класу в цілому, а не для окремого об'єкта. Для визначення застосовується оператор static . Наприклад :

```
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
```

У даному випадку визначено статичний метод nameToUpper(). Як параметр він приймає об'єкт Person та переводить його ім'я у верхній регістр. Оскільки статичний метод відноситься до класу в цілому , а не до об'єкта, то ми не можемо використовувати в ньому ключове слово this і через нього звертатися до властивостей об'єкт.

Практична частина

Завдання 1. Відповідно до свого варіанта визначити об'єкт, із заданими полями та методом виведення інформації про об'єкт. Об'єкт створюється за допомогою літералу.

1. Процесор
    a. Тактова частота
    b. Кількість ядер
    c. Розрядність процесора
2. Пам'ять
    a. Тип пам'яті
    b. Швидкість читання
    c. Швидкість запису
3. Корпус
    a. Потужність блоку живлення корпуси
    b. Форм-фактор материнської плати
    c. Охолодження
4. Твердий диск
    a. Тип накопичувача
    b. Об'єм
    c. Інтерфейс
    d. Швидкість обертання шпинделя
5. SSD диск
    a. Об'єм Інтерфейс

    b. Тип мікросхем
    c. Flash
    d. Контролер
6. Оптичний носій
    a. Тип установки
    b. Тип
    c. Інтерфейс підключення
    d. Максимальна швидкість читання
7. Клавіатура
    a. Тип
    b. Технологія перемикання
    c. Призначення
    d. Інтерфейс підключення
8. Миша
    a. Інтерфейс підключення
    b. Призначення
    c. Тип сенсора
    d. Модель сенсора
    e. Максимальна роздільна здатність сенсора
9. Звукова карта
    a. Аналогові виходи
    b. Аналогові входи
    c. Мікрофонні входи
    d. Тип
    e. Кількість каналів
    f. Інтерфейс підключення
10. Відеокарта
    a. Інтерфейс
    b. Виробник графічного процесора
    c. Графічний процесор
    d. Частота графічного процесора
    e. Відеопам'ять (обсяг)
11. Блок безперебійного
    a. Тип AVR
    b. Потужність
    c. Час автономного живлення
12. Планшет
    a. Діагональ екрану
    b. Процесор
    c. Роздільна здатність екрана
    d. Операційна система
    e. Оперативна пам'ять
    f. Внутрішня пам'ять
13. Електронна книга
    a. Розмір екрана
    b. Тип екрану

c. Роздільна здатність екрана
d. Флеш-пам'ять

14. Акустика
    a. Тип
    b. Підсилювач
    c. Максимальна потужність
    d. Матеріал корпусу
15. Фітнес-браслет
    a. Об'єм оперативної пам'яті
    b. Підтримувані платформи
    c. Датчики
16. Акумулятор
    a. Тип акумулятора
    b. Ємність
    c. Полярність
    d. Напруга
17. Мобільний телефон
    a. Тип
    b. Операційна система
    c. Розмір екрану
    d. Роздільна здатність екрана
    e. Оперативна пам'ять
18. Радіоприймач
    a. Живлення
    b. Тип
    c. Вихідна потужність звуку
    d. Тип тюнера
    e. Модуляція
19. Телевізор
    a. Тип
    b. Діагональ
    c. Частота
    d. Дозвіл
20. Фотоапарат
    a. Тип камери
    b. Тип матриці
    c. Розмір екрана
    d. Кількість точок матриці
    e. Максимальна роздільна здатність відео
21. Відеокамера
    a. Роздільна здатність відеозйомки
    b. Формат стиснення відео
    c. Розмір екрана
    d. Кількість точок матриці
22. Принтер
    a. Тип

b. Формат
c. Кількість кольорів
d. Технологія друку
e. Швидкість друку

23. Ігрова приставка
    a. Тип
    b. Об'єм накопичувача
    c. Максимальна роздільна здатність в іграх
24. IP камера
    a. Тип матриці
    b. Конструкція
    c. Стандарти безпровідного зв'язку
    d. Мережний інтерфейс
25. Портативна рація
    a. Стандарт
    b. Кількість каналів
    c. Дальність радіозв'язку
    d. Кількість радіостанцій у комплекті
26. Модем
    a. Тип
    b. Інтерфейс підключення
    c. Підтримка мобільного зв'язку
    d. Підтримка карт пам'яті
27. Монітор
    a. Діагональ
    b. Співвідношення сторін
    c. Роздільна здатність
    d. Частота оновлення екрану
28. Велосипед
    a. Клас
    b. Матеріал рами
    c. Тип трансмісії
    d. Діаметр коліс
29. Мікрохвильова піч
    a. Тип
    b. Об'єм
    c. Споживана потужність
    d. Вихідна потужність мікрохвиль
30. Пральна машина
    a. Завантаження білизни
    b. Максимальне завантаження
    c. Клас енергоспоживання
    d. Клас прання
    e. Максимальна швидкість віджиму

Завдання 2. Написати функцію-конструктор для створення об'єктів відповідно до варіанта. Визначити гетери та сеттери для отримання доступу до властивостей. У прототип об'єктів додати властивість - дата виходу ринку, метод відображення інформації про об'єкт.

Завдання 3. Описати клас, який описує об'єкти відповідно до варіанта. Передбачити конструктор з параметрами, гетери та сеттери для отримання доступу до властивостей, метод відображення інформації про об'єкт.

Завдання 4. Описати клас, що є спадкоємцем класу завдання 3. У похідному класі додати властивості: дата виходу ринку, вартість. Перевизначити метод виведення інформації про об'єкт.

Завдання 5. За підсумками розробленого класу завдання 4 створити масив об'єктів. Визначити об'єкти з максимальною та мінімальною вартістю, сумарну вартість усіх об'єктів, обчислити середню вартість об'єкта, підрахувати кількість об'єктів із вартістю вище за середню. Для виконання завдання використовувати вбудовані методи масивів та об'єкт Math.

Контрольні питання для захисту робіт:

1. Як створити об'єкт у JavaScript?
2. Що таке функція конструктор?
3. Що таке Prototype?
4. Як створити метод об'єкта?
5. Як змінити властивості об'єкта?
6. Як визначити власні властивості об'єкта?
7. Як додати, видалити властивість?
8. Як визначити клас?
9. Як визначити клас, успадкований від заданого?
10. Як звернутися до функціоналу базового класу з похідного?