**Державний вищий навчальний заклад**

**Ужгородський національний університет**

**Факультет інформаційних технологій**

**ЛАБОРАТОРНА РОБОТА № 9**

**Тема:** “JavaScript ч. 2”

Виконав студент І курсу

спеціальності «Інженерія

програмного забезпечення»

Романюк Артем Романович

**Ужгород-2024**

***Мета:*** поглибити знання з скриптової мови **JavaScript**.

***Завдання до роботи:***

1. Виконати домашні завдання описані у теоретичних матеріалах до лекцій 4,5,7.
2. Завдання в лекція 7 (ООП в JS) наступне:

- створити щонайменше 3 функції конструктори (класи), Можна застосовувати, навіть бажано, ES6 синтаксис

- проілюструвати на прикладі трьох і більше класах прототипне наслідування в JS.

- опишіть як ми можемо інтерпретувати (пояснити) значення вищезгаданого результату "[object Object]"

- чому [].toString() не виводить "[object Array]"

- реалізувати вивід "[object Array]" для масиву

1. Оформити звіт за взірцем, де представити результати код, скріншоти виконання, пояснення коду чи вирішення проблеми (якщо таке потрібно).

**Хід роботи:**

1. За допомогою цикла реалізую код, який виводить надану картинку

function Task1() {

const startLength = 9;

const maxSteps = 5;

for (let i = 1; i <= maxSteps; i++) {

console.log('\*'.repeat(startLength + i \* 2));

}

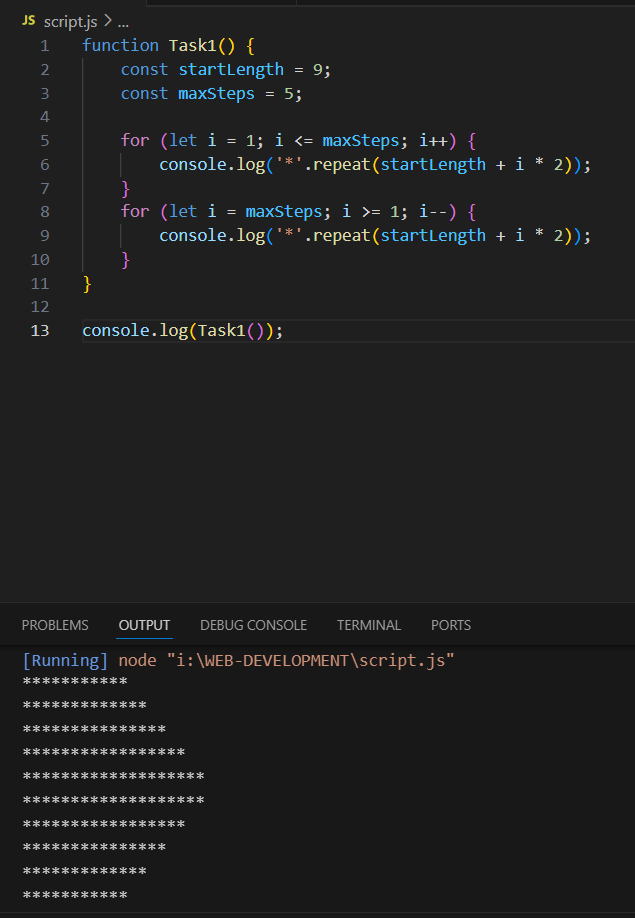
for (let i = maxSteps; i >= 1; i--) {

console.log('\*'.repeat(startLength + i \* 2));

}

}

console.log(Task1());



2. За допомогою циклу while / do-while реалізувати timer на 10 секунд.

function Task2() {

const endTime = Date.now() + 10000;

while (true) {

const remainingTime = endTime - Date.now();

if (remainingTime <= 0) {

console.log("Таймер завершився");

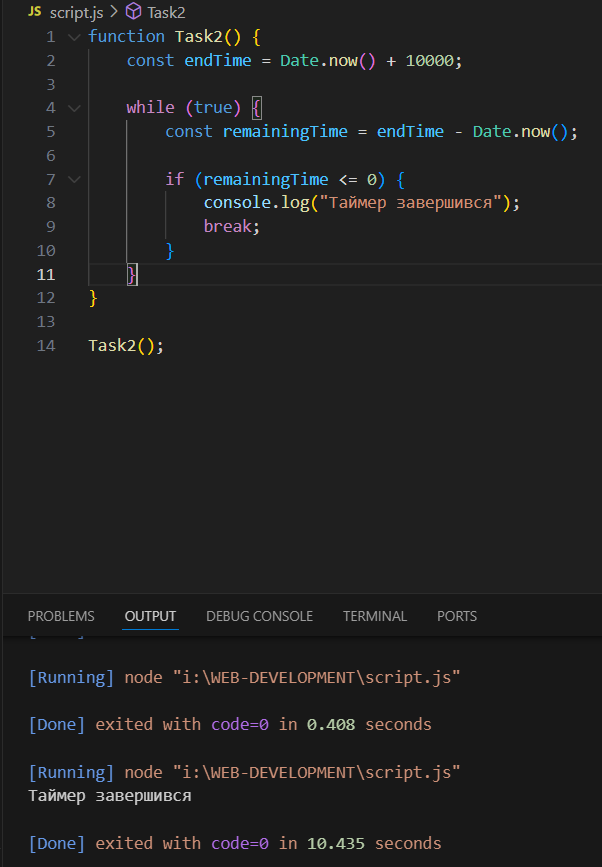
break;

}

}

}

Task2();



3. Використовуючи літеральну нотацію створюю об'єкт car з властивістю speedometer = 0;

const car ={

speedometer: 0

};

4. Використовуючи методи Object додаю до обєкту car наступні методи: setSpeedometer, що оновлює дані speedometer (сеттер) getSpeedometer, що повертає вміст speedometer (геттер) clearSpeedometr, що очищує вміст speedometer.

Object.defineProperty(car, "setSpeed",{

value: function (value){

this.speedometer = value;

return this;

}

});

Object.defineProperty(car, "getSpeed",{

value: function (){

console.log(this.speedometer)

return this;

}

});

Object.defineProperty(car,"clearSpeed",{

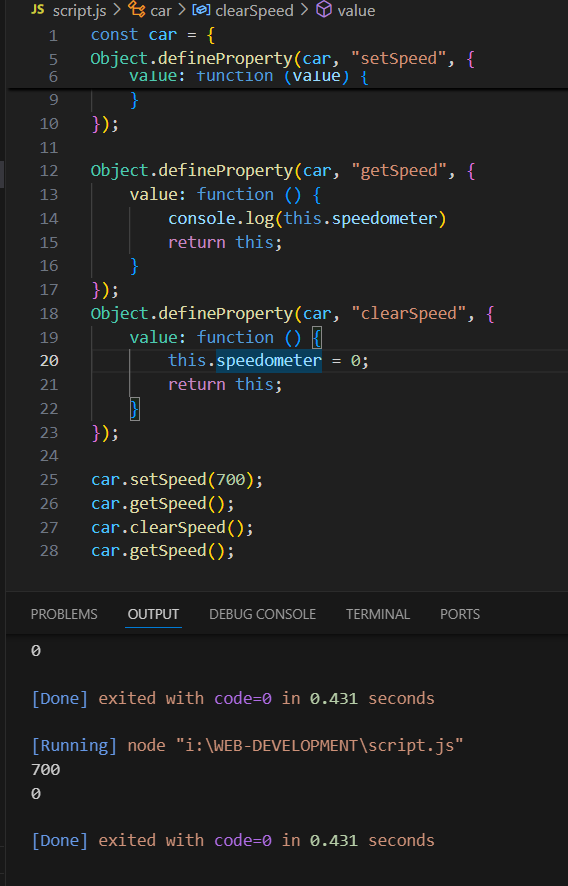
value: function (){

this.speedometer = 0;

return this;

}

});



5. Модифікую код таким чином, щоб можна було зробити наступне: car.setSpeed(200).setSpeed(300).getSpeed().clearSpeed() // Ланцюжковий виклик

Object.defineProperty(car, "setSpeed",{

value: function (value){

this.speedometer = value;

return this;

}

});

Object.defineProperty(car, "getSpeed",{

value: function (){

console.log(this.speedometer)

return this;

}

});

Object.defineProperty(car,"clearSpeed",{

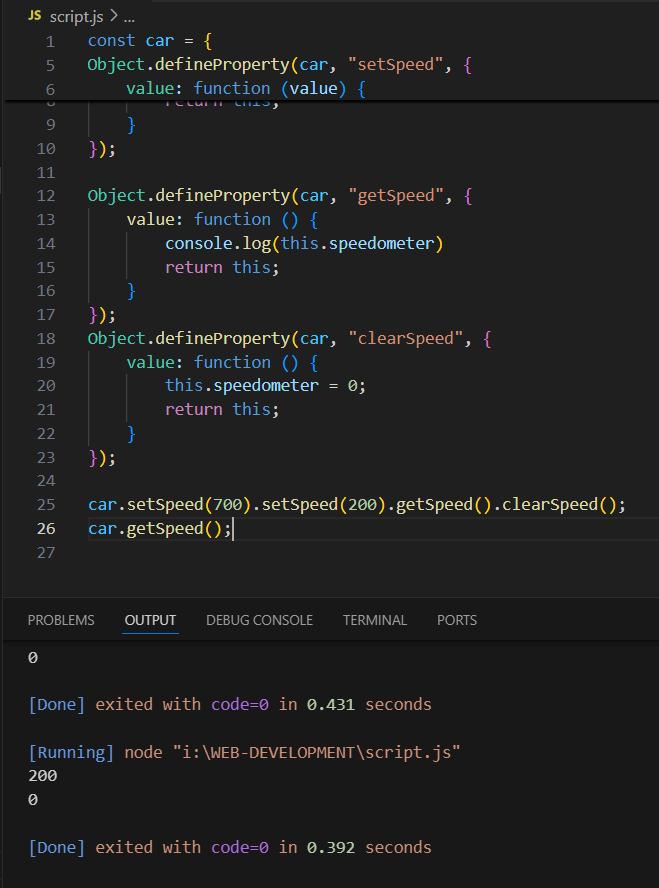
value: function (){

this.speedometer = 0;

return this;

}

});

****

6. Cтворюю 3 функції конструктори (класи), застосовуючи ES6 синтаксис

class Film {

constructor(title, author, times) {

this.title = title;

this.author = author;

this.times = times;

}

getSummary() {

return `${this.title} was directed by ${this.author} and continues ${this.times} hours`;

}

isLongFilm() {

return this.times > 3;

}

getWatchingTime(averageTime) {

return `${this.times / averageTime} hours`;

};

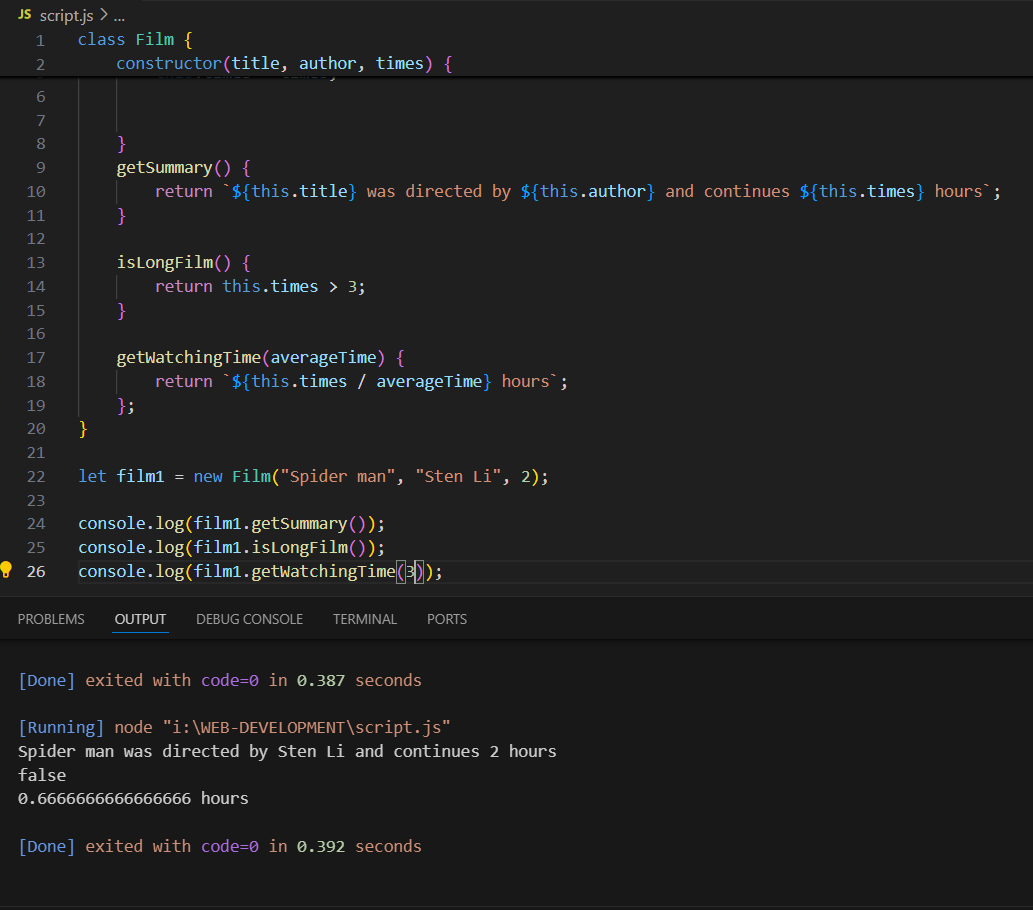
}

let film1 = new Film("Spider man", "Sten Li", 2);

console.log(film1.getSummary());

console.log(film1.isLongFilm());

console.log(film1.getWatchingTime(3));



class Student {

constructor(name, age, courses) {

this.name = name;

this.age = age;

this.courses = courses;

}

addCourse(newCourse) {

if (this.courses.includes(newCourse))

throw new Error("Студент вже записаний на цей курс!!");

this.courses.push(newCourse);

}

removeCourse(course) {

const index = this.courses.indexOf(course);

if (index > -1) {

this.courses.splice(index, 1);

} else {

throw new Error("Студент не записаний на такий курс");

}

}

isTakingCourse(course) {

return this.courses.includes(course);

}

GetSummary() {

return `Student ${this.name} is ${this.age} old and take this courses: ${this.courses}`

}

}

const Artem = new Student("Artem", "17", ["programming", "web", "physic"]);

Artem.isTakingCourse("programming")

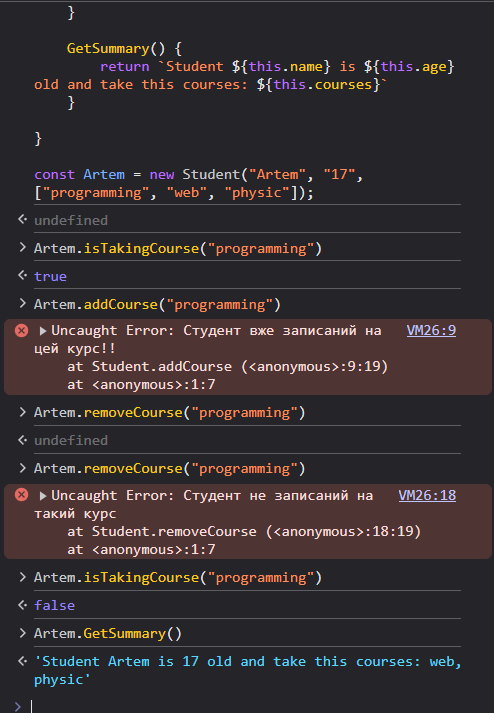
Artem.addCourse("programming")

Artem.removeCourse("programming")

Artem.removeCourse("programming")

Artem.isTakingCourse("programming")

Artem.GetSummary()



class Person {

constructor(name, age) {

this.name = name;

this.age = age;

}

isOld() {

return this.age >= 18;

}

introduce() {

return `Я ${this.name}, мені ${this.age} роки`

}

}

const Max = new Person("Максим", 24);

console.log(Max.isOld());

console.log(Max.introduce());



7. Проілюстровую на прикладі трьох і більше класах прототипне наслідування в JS.

class Person {

constructor(name, age) {

this.name = name;

this.age = age;

}

isOld() {

return this.age >= 18;

}

introduce() {

return `Я ${this.name}, мені ${this.age} роки`

}

}

class Teacher extends Person {

constructor(name, age, subject) {

super(name, age);

this.subject = subject;

}

teach() {

return `${this.name} вчить ${this.subject}`;

}

}

const buchuk = new Teacher("Роман Юрійович", 18, "веб-розробка");

console.log(buchuk.introduce());

console.log(buchuk.teach());

****

**class Person {**

**constructor(name, age) {**

**this.name = name;**

**this.age = age;**

**}**

**isOld() {**

**return this.age >= 18;**

**}**

**introduce() {**

**return `Я ${this.name}, мені ${this.age} роки`**

**}**

**}**

**class Child extends Person {**

**constructor(name, age, friends) {**

**super(name, age);**

**this.friends = friends;**

**}**

**getFriends() {**

**return `${this.name} дружить з ${this.friends.join(", ")}.`;**

**}**

**addFriend(friend) {**

**this.friends.push(friend);**

**}**

**}**

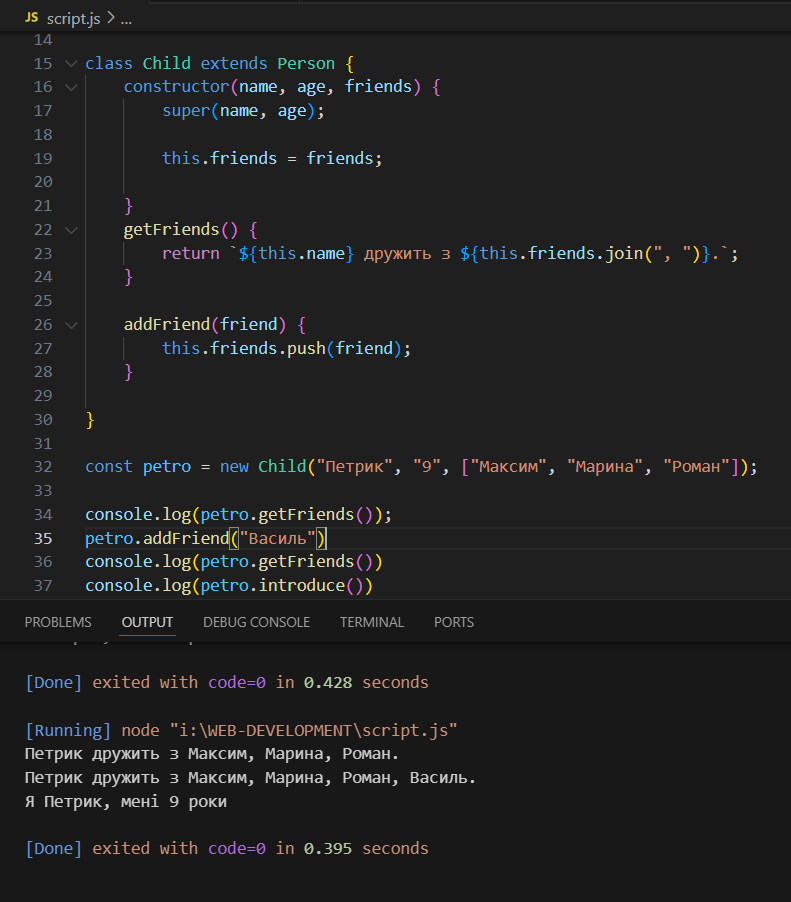
**const petro = new Child("Петрик", "9", ["Максим", "Марина", "Роман"]);**

**console.log(petro.getFriends());**

**petro.addFriend("Василь")**

**console.log(petro.getFriends())**

**console.log(petro.introduce())**

****

**class Person {**

**constructor(name, age) {**

**this.name = name;**

**this.age = age;**

**}**

**isOld() {**

**return this.age >= 18;**

**}**

**introduce() {**

**return `Я ${this.name}, мені ${this.age} роки`**

**}**

**}**

**class Child extends Person {**

**constructor(name, age, friends) {**

**super(name, age);**

**this.friends = friends;**

**}**

**getFriends() {**

**return `${this.name} дружить з ${this.friends.join(", ")}.`;**

**}**

**addFriend(friend) {**

**this.friends.push(friend);**

**}**

**}**

**class Teenager extends Child {**

**constructor(name, age, friends, school, grades) {**

**super(name, age, friends);**

**this.school = school;**

**this.grades = grades;**

**}**

**getSchool() {**

**return `${this.name} вчиться ${this.school}`;**

**}**

**getAverageGrade() {**

**let sum = 0;**

**for (let val of this.grades) {**

**sum += val;**

**}**

**return sum / this.grades.length;**

**}**

**}**

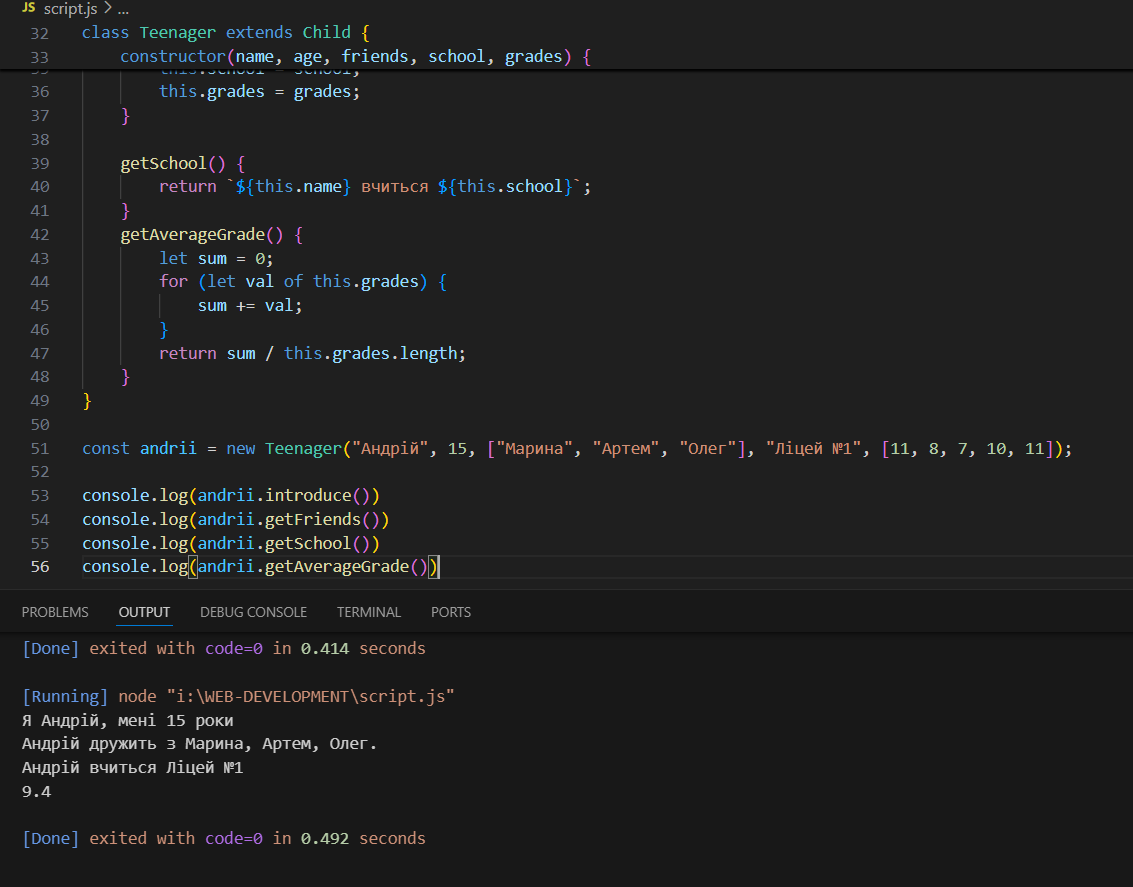
**const andrii = new Teenager("Андрій", 15, ["Марина", "Артем", "Олег"], "Ліцей №1", [11, 8, 7, 10, 11]);**

**console.log(andrii.introduce())**

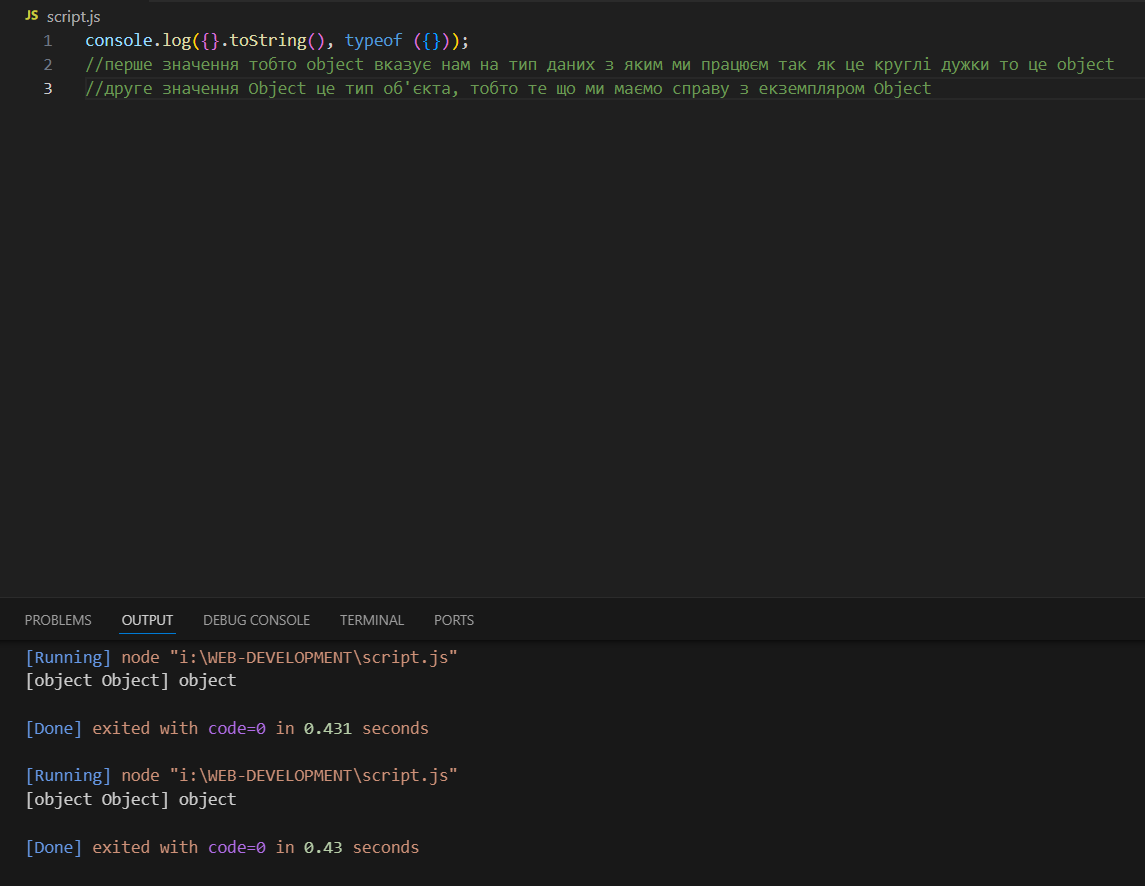
**console.log(andrii.getFriends())**

**console.log(andrii.getSchool())**

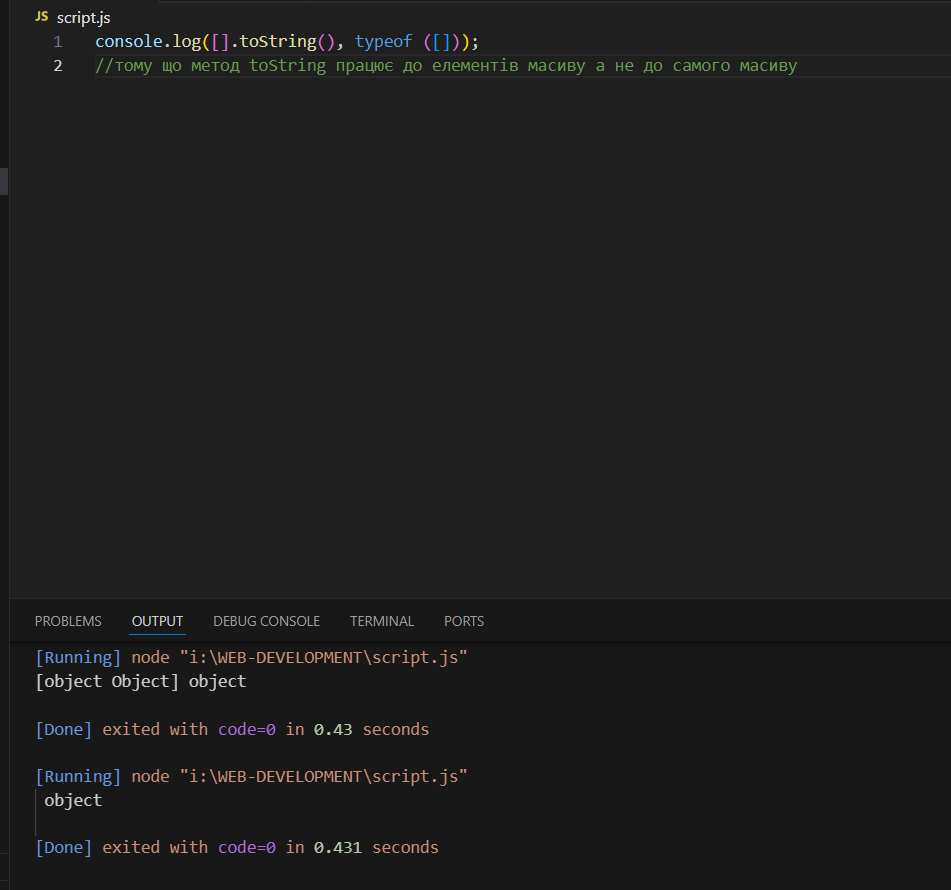
**console.log(andrii.getAverageGrade())**

****

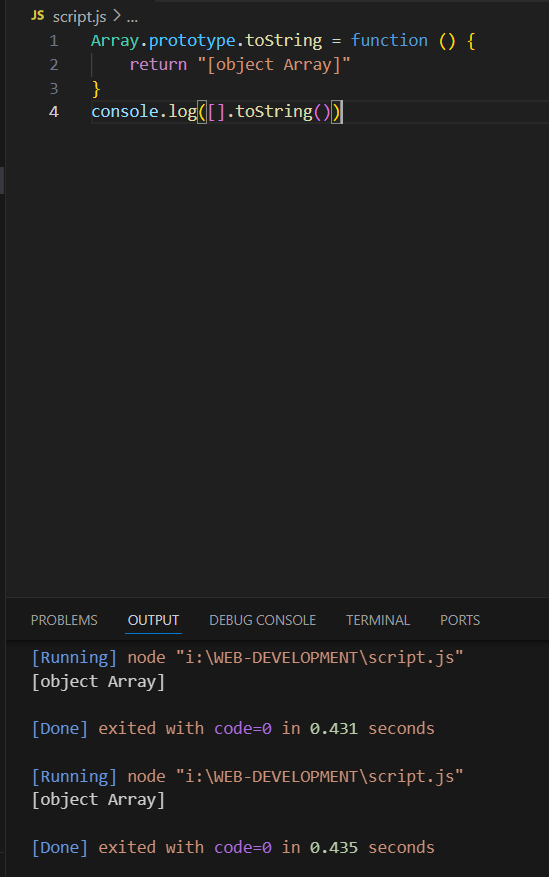
8. опишіть як ми можемо інтерпретувати (пояснити) значення вищезгаданого результату"[objectObject]"



9. чому [].toString() не виводить "[object Array]".



10. реалізувати вивід "[object Array]" для масиву.



**Висновок:**

На основі виконаної роботи можна зробити висновок, що лабораторна робота дозволила закріпити знання з об'єктно-орієнтованого програмування на мові JavaScript. Ми успішно створили та використали класи і прототипне наслідування, а також реалізували базові функції для роботи з об’єктами та масивами, що сприяло поглибленню розуміння структури даних і методів JavaScript.