ЛР 15-16 ООП в JavaScript

Порядок выполнения лабораторной работы:

1 Изучить краткие теоретические сведения по теме работы.

Ответить на контрольные вопросы:

Тема "Объекты в Java Script"

1. Что понимается под объектом в Java Script? Приведите примеры известных вам объектов в Java Script.

В JavaScript объект — это коллекция свойств, где каждое свойство состоит из ключа (строка) и значения. Примеры встроенных объектов: Math, Date, Array, Object.

2. Приведите пример описания объекта в JS.

3. Приведите пример доступа к свойствам объекта в JS

4. Какая команда в JS позволяет выполнить удаление свойства. Приведите пример.

5. Какая команда в JS позволяет выполнить проверку существования свойства. Приведите пример.

6. Какая команда в JS позволяет выполнить перебор свойств объекта. Приведите пример.

Тема "ООП в Java Script"

1 В чем заключается суть ООП программирования и его отличие от традиционного процедурного программирования?

ООП (Объектно-Ориентированное Программирование) фокусируется на создании объектов, которые инкапсулируют данные и методы для работы с этими данными. Это отличается от процедурного программирования, где основной фокус на функциях и их последовательном выполнении.

2 Что понимается под Классом и Объектом в JS?

Класс — это шаблон для создания объектов, содержащий свойства и методы. Объект — это экземпляр класса.

3. Приведите пример синтаксиса класса в JS.

4 Как осуществляется создание объектов класса? Приведите пример.

5 Как происходит обращение к полям и методам внутри класса?

6 Как обозначаются в JS публичные, приватные и защищённые поля и методы?

Публичные поля и методы обозначаются без специального символа.

Приватные поля и методы обозначаются с использованием # перед именем (например, #privateField).

Защищённых полей в JS нет, но можно использовать соглашения об именовании (например, \_protectedField).

7 Для чего в ООП в JS используется служебное слово static?

static обозначает, что метод или свойство принадлежит классу, а не экземпляру класса. Они могут быть вызваны без создания объекта класса.

8 Что такое геттеры и сеттеры ? Для чего и как они используются в ООП в JS?

Геттеры и сеттеры — это методы, которые позволяют получить или изменить значения свойств объекта.

9. Что понимается под базовыми принципами ООП: инкапсуляция, наследование и полиморфизм?

Инкапсуляция — это скрытие внутренней реализации объекта от внешнего мира. Наследование позволяет создавать новые классы на основе существующих. Полиморфизм позволяет использовать один интерфейс для разных типов объектов.

10. На каждый принцип подберите в сети Интернет примеры реализации базовых принципов ООП на ЯП JS: инкапсуляция, наследование, полиморфизм.

Код:

// Тема "Объекты в JavaScript"

// 2 Пример описания объекта в JS:

let person = {

    name: "John",

    age: 30,

    greet: function () {

        console.log("Hello!");

    }

};

// 3 Пример доступа к свойствам объекта:

console.log(person.name);

console.log(person["age"]);

// 4 Удаление свойства:

delete person.age;

// 5 Проверка существования свойства:

console.log("name" in person); // true

console.log(person.hasOwnProperty("age")); // false

// 6 Перебор свойств объекта:

for (let key in person) {

    console.log(key + ": " + person[key]);

}

// Тема "ООП в JavaScript"

// 3 Пример синтаксиса класса:

class Person {

    constructor(name, age) {

        this.name = name;

        this.age = age;

    }

    greet() {

        console.log("Hello, my name is " + this.name);

    }

}

// 4 Создание объектов класса:

let john = new Person("John", 30);

john.greet(); // Hello, my name is John

// 5 Обращение к полям и методам:

class Employee {

    constructor(name, position) {

        this.name = name;

        this.position = position;

    }

    introduce() {

        console.log("Hi, I'm " + this.name + " and I work as a " + this.position);

    }

}

let alice = new Employee("Alice", "developer");

alice.introduce(); // Hi, I'm Alice and I work as a developer

// 8 Геттеры и сеттеры:

class Car {

    constructor(brand) {

        this.\_brand = brand;

    }

    get brand() {

        return this.\_brand;

    }

    set brand(value) {

        this.\_brand = value;

    }

}

let myCar = new Car("Tesla");

console.log(myCar.brand); // Tesla

myCar.brand = "BMW";

console.log(myCar.brand); // BMW

// 10 Примеры реализации базовых принципов ООП:

// Инкапсуляция:

class Product {

    #price; // Приватное поле

    constructor(name, price) {

        this.name = name;

        this.#price = price;

    }

    getPrice() {

        return this.#price;

    }

}

let product = new Product("Laptop", 1500);

console.log(product.getPrice()); // 1500

// Наследование:

class Animal {

    speak() {

        console.log("Animal speaks");

    }

}

class Dog extends Animal {

    speak() {

        console.log("Woof!");

    }

}

let dog = new Dog();

dog.speak(); // Woof!

// Полиморфизм:

class Bird {

    speak() {

        console.log("Tweet");

    }

}

class DogPolymorph {

    speak() {

        console.log("Woof");

    }

}

function makeItSpeak(animal) {

    animal.speak();

}

makeItSpeak(new Bird()); // Tweet

makeItSpeak(new DogPolymorph());  // Woof

ПРАКТИЧЕСКАЯ ЧАСТЬ РАБОТЫ

Задание 1

На каждый изучаемый вопрос из теории по ООП подберите в электронных учебниках пример реализации. Отработайте его в редакторе кода. Продемонстрируете результаты выполненной работы преподавателю.

Задание 2

Применить изученные принципы ООП на практике на примере создания любого ООП приложения (браузерная игра, мини-игра на JS , и ТД).

CSS:

JS:

class WordGuessGame {

    constructor() {

        this.secretWord = 'ФЫВ';

        this.attempts = 6;

        this.guessInput = document.getElementById('guess-input');

        this.guessButton = document.getElementById('guess-button');

        this.resultElement = document.getElementById('result');

        this.attemptsElement = document.getElementById('attempts');

        this.guessButton.addEventListener('click', this.handleGuess.bind(this));

    }

    handleGuess() {

        const guess = this.guessInput.value.toUpperCase();

        this.guessInput.value = '';

        if (guess === this.secretWord) {

            this.resultElement.hidden = false;

            this.resultElement.innerText = 'Результат: Вы угадали слово!';

            this.guessButton.disabled = true;

        } else {

            this.attempts--;

            this.attemptsElement.innerText = `Осталось попыток: ${this.attempts}`;

            if (this.attempts === 0) {

                this.resultElement.innerText = `Результат: Секретное слово - ${this.secretWord}`;

                this.guessButton.disabled = true;

            } else {

                this.resultElement.innerText = 'Результат: Вы не угадали слово. Попробуйте еще раз!';

            }

        }

    }

}

const game = new WordGuessGame();