**Лабораторная работа 15-16**

Объекты в Java Script. ООП в Java Script

Контрольные вопросы:

1. Что понимается под объектом в Java Script?

Объект в JavaScript — это коллекция свойств, где каждое свойство представляет собой пару "ключ-значение". Ключи (или имена свойств) являются строками, а значения могут быть любого типа: числа, строки, функции и даже другие объекты.

1. Приведите пример описания объекта в JS.

const person = {

name: "Alice",

age: 30,

job: "Developer"

};

1. Приведите пример доступа к свойствам объекта в JS.

console.log(person.name); // "Alice"

console.log(person["age"]); // 30

1. Какая команда в JS позволяет выполнить удаление свойства.

Для удаления свойства используется оператор delete:

delete person.job;

1. Какая команда в JS позволяет выполнить проверку существования свойства.

Для проверки существования свойства можно использовать оператор in или метод hasOwnProperty:

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

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

1. Какая команда в JS позволяет выполнить перебор свойств объекта.

Для перебора свойств объекта можно использовать цикл for...in:

for (let key in person) {

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

}

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

Суть ООП заключается в организации кода вокруг объектов, которые объединяют данные и методы, действующие над этими данными. Это позволяет создавать более модульный и гибкий код. В отличие от процедурного программирования, где акцент делается на последовательности выполнения функций, ООП фокусируется на взаимодействии объектов.

1. Что понимается под Классом и Объектом в JS?

Класс — это шаблон для создания объектов, который определяет их свойства и методы. Объект — это конкретный экземпляр класса.

1. Приведите пример синтаксиса класса в JS.
2. Как осуществляется создание объектов класса? Приведите пример.

Создание объекта класса осуществляется с помощью оператора new:

const alice = new Person("Alice", 30);

alice.greet(); // "Hello, my name is Alice"

1. Как происходит обращение к полям и методам внутри класса?

class Person {

constructor(name) {

this.name = name;

}

greet() {

console.log(Hello, my name is ${this.name});

}

}

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

- Публичные поля и методы не имеют специального обозначения.

- Приватные поля обозначаются символом #, например: #privateField.

- Защищённые поля не имеют строгого синтаксиса, но обычно используются соглашения об именовании (например, \_protectedField).

1. Для чего в ООП в JS используется служебное слово static?

Служебное слово static используется для определения статических методов и полей, которые принадлежат самому классу, а не его экземплярам. Они могут вызываться без создания экземпляра класса.

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

Геттеры и сеттеры — это специальные методы для доступа к свойствам объекта. Геттер позволяет получить значение свойства, а сеттер — установить его:

class Person {

constructor(name) {

this.\_name = name;

}

get name() {

return this.\_name;

}

set name(newName) {

this.\_name = newName;

}

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

- Инкапсуляция — это сокрытие внутреннего состояния объекта и предоставление доступа к нему только через публичные методы.

- Наследование — это механизм, позволяющий создавать новый класс на основе существующего, заимствуя его свойства и методы.

- Полиморфизм — это возможность использовать один интерфейс для работы с различными типами объектов.

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

- Инкапсуляция:

class BankAccount {

#balance; // приватное поле

constructor(initialBalance) {

this.#balance = initialBalance;

}

deposit(amount) {

this.#balance += amount;

}

getBalance() {

return this.#balance;

}

}

- Наследование:

class Animal {

speak() {

console.log("Animal speaks");

}

}

class Dog extends Animal {

speak() {

console.log("Woof!");

}

}

- Полиморфизм:

function makeAnimalSpeak(animal) {

animal.speak();

}

const dog = new Dog();

makeAnimalSpeak(dog); // "Woof!"