**Лабораторная работа №15-16**

**ООП в JavaScript**

Тема "Объекты в Java Script"

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

В JavaScript объектом считается нечто, что представляет собой набор связанных данных и/или функциональности. Объект может содержать свойства (данные) и методы (функции). Объекты являются основными структурами данных в JavaScript и могут представлять практически любые сущности.

Примеры известных объектов в JavaScript:

- Math – математические функции и константы.

- Date – работа с датами и временем.

- Array – работа с массивами.

- JSON – методы для работы с JSON-форматом.

- Window – объект, представляющий окно вашего браузера.

2 Приведите пример описания объекта в JS.

Пример описания простого объекта:

const person = {

name: "Иван",

age: 30,

isStudent: false,

greet: function() {

console.log("Привет, меня зовут " + this.name);

}

};

3 Приведите пример доступа к свойствам объекта в JS.

Доступ к свойствам объекта можно осуществлять двумя способами: через точечную нотацию или через квадратные скобки.

Пример с точечной нотацией:

console.log(person.name); // Вывод: Иван

Пример с квадратными скобками:

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

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

Для удаления свойства из объекта используется оператор delete.

Пример:

delete person.isStudent;

console.log(person); // Вывод: {name: "Иван", age: 30, greet: f}

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

Для проверки существования свойства в объекте можно использовать оператор in или метод hasOwnProperty.

Пример с оператором in:

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

console.log("isStudent" in person); // Вывод: false

Пример с методом hasOwnProperty:

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

console.log(person.hasOwnProperty("isStudent")); // Вывод: false

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

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

Пример:

for (const key in person) {

if (person.hasOwnProperty(key)) {

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

}

}

Вывод:

name: Иван

age: 30

greet: function() { ... }

Этот цикл перебирает все собственные свойства объекта (включая его методы)

Тема "ООП в Java Script"

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

Суть объектно-ориентированного программирования (ООП) заключается в моделировании реальных объектов и их взаимодействий через классы и объекты. Основные аспекты ООП включают инкапсуляцию, наследование и полиморфизм. ООП позволяет группировать связанные данные и функции в единое целое, что улучшает структуру и читаемость кода.

Основные отличия от традиционного процедурного программирования:

- Структура кода: В ООП код организован в классы и объекты, а в процедурном - в функции и процедуры.

- Инкапсуляция: В ООП данные защищены от непосредственного доступа и изменений извне, тогда как в процедурном доступ ко всем данным свободный.

- Наследование и полиморфизм: позволяют создавать иерархии объектов и переопределять методы, чего нет в процедурном программировании.

2 Что понимается под Классом и Объектом в JS?

- Класс – это шаблон, который описывает свойства и методы, которые объекты этого класса могут иметь. Это определение структуры объекта.

- Объект – это конкретный экземпляр класса, который содержит реальные значения свойств и может использовать методы, описанные в классе.

3 Приведите пример синтаксиса класса в JS.

Пример синтаксиса класса в JavaScript:

class Animal {

constructor(name, species) {

this.name = name;

this.species = species;

}

makeSound() {

console.log(`${this.name} издает звук.`);

}

}

4 Как осуществляется создание объектов класса? Приведите пример.

Создание объектов класса осуществляется с помощью оператора new.

Пример:

const dog = new Animal('Шарик', 'Собака');

dog.makeSound(); // Вывод: Шарик издает звук.

5 Как происходит обращение к полям и методам внутри класса?

Обращение к полям и методам внутри класса осуществляется через ключевое слово this, что ссылается на текущий экземпляр класса.

Пример:

class Animal {

constructor(name) {

this.name = name;

}

greet() {

console.log(`Привет, я ${this.name}!`);

}

}

const cat = new Animal('Мурка');

cat.greet(); // Вывод: Привет, я Мурка!

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

- Публичные поля и методы: они доступны извне класса и не требуют каких-либо специальных обозначений.

- Приватные поля и методы: обозначаются символом # перед именем свойства или метода.

- Защищенные поля и методы: в JavaScript нет встроенной поддержки защищенных полей, но класс может использовать соглашение об именах (например, префикс \_) для обозначения защищенных свойств.

Пример:

class Example {

publicMethod() {

console.log('Это публичный метод');

}

#privateMethod() {

console.log('Это приватный метод');

}

}

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

Ключевое слово static используется для определения статических методов и свойств, которые принадлежат самому классу, а не его экземплярам. Они могут быть вызваны без создания экземпляра класса.

Пример:

class MathUtil {

static add(x, y) {

return x + y;

}

}

console.log(MathUtil.add(5, 3)); // Вывод: 8

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

Геттеры и сеттеры предназначены для управления доступом к полям объекта. Геттер позволяет получать значение свойства, а сеттер — изменять его.

Пример:

class Person {

constructor(name) {

this.\_name = name;

}

get name() {

return this.\_name;

}

set name(newName) {

this.\_name = newName;

}

}

const person = new Person('Иван');

console.log(person.name); // Вывод: Иван

person.name = 'Петр';

console.log(person.name); // Вывод: Петр

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

- Инкапсуляция: это сокрытие внутренних состояний и поведения объекта от внешнего мира, предоставляя интерфейс для взаимодействия. Это помогает защитить данные и управлять доступом к ним.

- Наследование: это механизм, который позволяет одному классу наследовать свойства и методы другого класса. Он способствует повторному использованию кода и созданию иерархий классов.

- Полиморфизм: это способность объектов разных классов отвечать на одни и те же сообщения (вызов методов) разными способами. Это позволяет создавать гибкие и расширяемые системы.

10. Примеры реализации базовых принципов ООП на ЯП JS.

Инкапсуляция:

class BankAccount {

constructor(balance) {

this.\_balance = balance; // приватное поле

}

deposit(amount) {

this.\_balance += amount;

}

getBalance() {

return this.\_balance;

}

}

const account = new BankAccount(100);

account.deposit(50);

console.log(account.getBalance()); // Вывод: 150

Наследование:

class Animal {

speak() {

console.log('Я животное');

}

}

class Dog extends Animal {

speak() {

console.log('Гав!');

}

}

const dog = new Dog();

dog.speak(); // Вывод: Гав!

Полиморфизм:

class Bird {

fly() {

console.log('Птица летит');

}

}

class Plane {

fly() {

console.log('Самолет летит');

}

}

function letFly(object) {

object.fly(); // Полиморфный вызов

}

letFly(new Bird()); // Вывод: Птица летит

letFly(new Plane()); // Вывод: Самолет летит