**ЛР 15-16 ООП в JavaScript**

**Контрольные вопросы**

1. **Что понимается под объектом в JavaScript? Приведите примеры известных вам объектов в JavaScript.**

Объект в JavaScript — это коллекция свойств, где каждое свойство состоит из пары "ключ: значение". Значением свойства может быть функция, которую называют методом объекта. Примеры объектов в JavaScript включают встроенные объекты, такие как `Math`, `Date`, `Array`, а также пользовательские объекты, созданные разработчиками.

2. **Приведите пример описания объекта в JS.**

const person = {

name: "John",

age: 30,

isEmployed: true,

greet: function() {

console.log("Hello!");

}

};

3. **Приведите пример доступа к свойствам объекта в JS.**

Доступ к свойствам объекта можно получить с помощью точечной или скобочной нотации:

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

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

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

Для удаления свойства используется оператор delete:

delete person.isEmployed;

console.log(person.isEmployed); // undefined

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

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

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

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

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

for (let key in person) {

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

}

// Вывод:

// name: John

// age: 30

// greet: function() { console.log("Hello!"); }

1. **Суть ООП программирования и его отличие от традиционного процедурного программирования**:

- ООП (Объектно-ориентированное программирование): Основывается на концепции объектов, которые содержат данные и методы для работы с этими данными. Основные принципы ООП включают инкапсуляцию, наследование и полиморфизм.

- Процедурное программирование: Основывается на последовательности процедур или функций, которые выполняются одна за другой. Код структурируется в виде функций, которые манипулируют данными.

2. **Класс и Объект в JS**:

- Класс: Это шаблон для создания объектов, который определяет их свойства и методы.

- Объект: Это экземпляр класса, содержащий конкретные данные и методы, определенные в классе¹.

3. **Пример синтаксиса класса в JS**:

class Rectangle {

constructor(height, width) {

this.height = height;

this.width = width;

}

calcArea() {

return this.height \* this.width;

}

}

4. **Создание объектов класса**:

const square = new Rectangle(10, 10);

console.log(square.calcArea()); // 100

5. **Обращение к полям и методам внутри класса**:

- Для обращения к полям и методам внутри класса используется ключевое слово this, которое указывает на текущий объект.

class Rectangle {

constructor(height, width) {

this.height = height;

this.width = width;

}

calcArea() {

return this.height \* this.width;

}

}

6. **Публичные, приватные и защищённые поля и методы в JS**:

- Публичные: Доступны везде.

- Приватные: Обозначаются символом `#` и доступны только внутри класса.

class Person {

#name;

constructor(name) {

this.#name = name;

}

getName() {

return this.#name;

}

}

const person = new Person("John");

console.log(person.getName()); // John

7. **Служебное слово `static` в ООП в JS**:

- Используется для определения статических методов и свойств, которые принадлежат классу, а не его экземплярам.

class MathUtils {

static square(number) {

return number \* number;

}

}

console.log(MathUtils.square(4)); // 16

8. **Геттеры и сеттеры в ООП в JS**:

- Геттеры: Используются для получения значения свойства.

- Сеттеры: Используются для установки значения свойства¹².

class User {

constructor(name) {

this.\_name = name;

}

get name() {

return this.\_name;

}

set name(value) {

this.\_name = value;

}

}

const user = new User("Alice");

console.log(user.name); // Alice

user.name = "Bob";

console.log(user.name); // Bob

9. **Базовые принципы ООП: инкапсуляция, наследование и полиморфизм**:

- Инкапсуляция: Сокрытие деталей реализации объекта от пользователя.

- Наследование: Создание новых классов на основе существующих.

- Полиморфизм: Использование различных объектов с одинаковыми методами.

10. **Примеры реализации базовых принципов ООП на ЯП JS**:

**Инкапсуляция:**

class Person {

#name;

constructor(name) {

this.#name = name;

}

getName() {

return this.#name;

}

}

const person = new Person("John");

console.log(person.getName()); // John

**Наследование:**

class Animal {

constructor(name) {

this.name = name;

}

speak() {

console.log(`${this.name} makes a noise.`);

}

}

class Dog extends Animal {

speak() {

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

}

}

const dog = new Dog("Rex");

dog.speak(); // Rex barks.

**Полиморфизм:**

class Bird {

fly() {

console.log("Bird is flying");

}

}

class Eagle extends Bird {

fly() {

console.log("Eagle is flying high");

}

}

class Penguin extends Bird {

fly() {

console.log("Penguin can't fly");

}

}

const birds = [new Bird(), new Eagle(), new Penguin()];

birds.forEach(bird => bird.fly());

// Bird is flying

// Eagle is flying high

// Penguin can't fly