ЛР 15-16 ООП в JavaScript

Тема "Объекты в JavaScript"

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

В JavaScript объект – это коллекция свойств, где каждое свойство содержит пару "ключ-значение". Объекты могут содержать данные (свойства) и действия (методы). Примеры встроенных объектов в JavaScript: `Object`, `Array`, `Date`, `Math`, `String`, `Function`.

2. Приведите пример описания объекта в JS.

```javascript

const person = {

name: "John",

age: 30,

greet: function() {

console.log("Hello!");

}

};

```

3. Приведите пример доступа к свойствам объекта в JS.

```javascript

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

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

```

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

Оператор `delete` используется для удаления свойства объекта.

```javascript

delete person.age;

console.log(person.age); // Вывод: undefined

```

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

Оператор `in` позволяет проверить, существует ли свойство в объекте.

```javascript

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

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

```

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

Для перебора свойств объекта используется цикл `for...in`.

```javascript

for (let key in person) {

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

}

```

Тема "ООП в JavaScript"

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

Основная идея объектно-ориентированного программирования (ООП) – моделирование реального мира через объекты, которые содержат данные и методы для работы с этими данными. В отличие от процедурного программирования, где программы строятся из функций, ООП направлено на создание объектов и их взаимодействие. ООП помогает упростить поддержку и расширяемость кода за счёт базовых принципов: инкапсуляции, наследования и полиморфизма.

2. Что понимается под Классом и Объектом в JS?

Класс – это шаблон для создания объектов с заданными свойствами и методами. Объект – это экземпляр класса, представляющий конкретную реализацию класса с уникальными значениями свойств.

3. Приведите пример синтаксиса класса в JS.

```javascript

class Animal {

constructor(name) {

this.name = name;

}

speak() {

console.log(this.name + " издает звук.");

}

}

```

4. Как осуществляется создание объектов класса? Приведите пример.

Объекты создаются с помощью оператора `new`.

```javascript

const dog = new Animal("Собака");

```

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

Для обращения к полям и методам внутри класса используется `this`. Например, `this.name` обращается к полю `name` текущего экземпляра класса.

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

В JS публичные поля и методы доступны для внешнего кода и записываются без символов. Приватные обозначаются символом `#` перед именем поля/метода, и доступны только внутри класса. Защищённые поля и методы (с возможностью наследования) не имеют специального синтаксиса, но соглашение — начинать имя с `\_`.

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

`static` обозначает метод или свойство, принадлежащее классу, а не конкретному объекту. Такие методы вызываются на самом классе, а не на экземпляре.

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

Геттеры и сеттеры – это методы для получения и установки значений свойств. Они помогают управлять доступом к свойствам, позволяя добавлять логику при чтении или записи значений.

```javascript

class Person {

constructor(name) {

this.\_name = name;

}

get name() {

return this.\_name;

}

set name(newName) {

this.\_name = newName;

}

}

```

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

- Инкапсуляция: Сокрытие деталей реализации объекта, предоставляя только необходимые интерфейсы.

- Наследование: Возможность класса "наследовать" свойства и методы другого класса.

- Полиморфизм: Способность объектов с одинаковым интерфейсом вести себя по-разному.

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

Пример кода для:

- Инкапсуляции: создание приватных свойств и методов с помощью символа `#`.

- Наследования: использование `extends` для создания подклассов.

- Полиморфизма: создание методов с одинаковыми именами в разных классах, но с разной реализацией.