1. Опишите, что такое инкапсуляция, наследование и полиморфизм. Приведите примеры их применения в коде.

Инкапсуляция - это принцип ООП, который ограничивает доступ к некоторым компонентам объекта и позволяет скрыть внутреннее устройство объекта от внешнего мира. Это делается через определение приватных и публичных методов и свойств.

class Animal {

  constructor(*name*) {

    this.name = name;

  }

  #speak() {

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

  }

}

const animal = new Animal('Rex');

animal.speak(); *//animal.speak is not a function*

function createCounter() {

  let count = 0; *// приватная переменная*

  function increment() { *// доступная функция с замыканием count*

    count++;

    console.log(count);

  }

  return {

    increment,

  };

}

const counter = createCounter();

counter.increment(); *// 1*

counter.increment(); *// 2*

console.log(counter.count); *// Undefined. Узел недоступен извне замыкания*

Наследование позволяет создавать новый класс на основе существующего. Это позволяет унаследовать методы и свойства одного класса в другом.

class Animal {

  constructor(*name*) {

    this.name = name;

  }

  speak() {

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

  }

}

class Dog extends Animal {

  speakDog() {

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

  }

}

const dog = new Dog('Rex');

dog.speak(); *// Rex makes a noise.*

dog.speakDog()*//Rex barks.*

Полиморфизм — это один из ключевых принципов объектно-ориентированного программирования (ООП), который позволяет использовать один и тот же интерфейс для разных типов объектов. Основная идея полиморфизма заключается в том, что методы могут работать с объектами различных классов, выглядящих как объекты одного класса через общий интерфейс или суперкласс.

Идея заключается в способности вызывать один и тот же метод для разных объектов, и при этом каждый объект реагирует по-своему.  
Чтобы это произошло полиморфизм использует наследование.

class Animal {

  speak() {

    console.log("The animal makes a sound.");

  }

}

class Cat extends Animal {

  speak() {

    console.log("The cat meows.");

  }

}

class Dog extends Animal {

  speak() {

    console.log("The dog barks.");

  }

}

const animals = [new Animal(), new Cat(), new Dog()];

animals.forEach(*animal* => animal.speak());

*// The animal makes a sound.*

*// The cat meows.*

*// The dog barks.*

2. Восстановление предыдущей версии кода в git

- Используйте `git log` для просмотра истории коммитов и получения SHA нужного вам коммита.

- Используйте `git checkout <commit-hash>` для временной "перемотки" на нужный коммит. Это не изменит ваш текущий бранч.

- Используйте `git reset --hard <commit-hash>` для отката текущего бранча на указанную версию. Это изменит историю коммитов.

- Если вы хотите отменить изменения, сделанные в последнем коммите, можно использовать `git revert <commit-hash>`, что создаст новый коммит, отменяющий

изменения.

3. Проверка строки на палиндром

function palinTest(*str*) {

  const cleanedStr = str.toLowerCase().replace(/[\W\_]/g, ''); *// Удаляем пробелы и спецсимволы*

  return cleanedStr === cleanedStr.split('').reverse().join('');;

}

console.log(palinTest("Ruot, tour")); *// true*

console.log(palinTest("Hello world")); *// false*

4 Опыт в IT и проекты на GitHub

Недавний проект(фриланс): Автоматизированная система расчета сырья и контроля транспортных средств  
- Цель: Разработал комплексное решение для расчета сырья, контроля транспортных средств по номерным знакам и автоматизированной обработки документов.  
- Управление заказами: Создана система с отдельными базами данных для транспортных средств и различными уровнями доступа пользователей.  
- Логирование и мониторинг: Реализована автоматическая регистрация загрузок транспортных средств с функцией экспорта в Excel для периодического мониторинга.

Тестовые задания на работу с реббит, обработка большого файла с ограничениями на оперативную память.

[Гитхаб](https://github.com/Ranky1ne?tab=repositories)