1) Write a js program to demonstrate the difference between global ,local & block scope ?

Answer:

Global Scope:A variable that is accessible everywhere in the program.

Local Scope: A variable that is declared inside the showScopes function, so it's only accessible within that function.

Block Scope:A variable that is declared inside the if block and is only accessible within that block. Outside the block, it’s undefined.

Example:

```
let schoolName = "Nightingales English High School";

function classroom() {
  // Local Scope: Only the teacher knows the subject
  let subject = "Mathematics";

    console.log(`Welcome to ${schoolName}`); // Accessing global scope
    console.log(`Today's class is: ${subject}`); // Accessing local scope

  if (true) {
    // Block Scope: The teacher announces the class rule only inside the class
    let classRule = "No talking during the teaching";
    console.log(`Class rule: ${classRule}`); // Accessing block scope
  }

  // Trying to access block variable outside the block
  console.log(typeof classRule); // Outputs "undefined"
}
```

```
// Accessing global variable
console.log(`School Name: ${schoolName}`); // Accessible

// Trying to access local variable in global scope
console.log(typeof subject); // Outputs "undefined"

classroom();
```

2)Create a closure that adds a number to a predefined value.Test it with different values ?

Answer:

A **closure** allows an inner function to access variables from its outer function even after the outer function has finished executing.

Example:

```
// Function that creates a closure
function createAdder(predefinedValue) {
    // Inner function (closure) that adds a number to the predefined value
   return function(number) {
       return predefinedValue + number;
   };
}

// Create closures with different predefined values
const addFive = createAdder(5);
const addTen = createAdder(10);

// Test the closures with different numbers
console.log(addFive(3));  // Output: 8 (5 + 3)
console.log(addFive(10)); // Output: 15 (5 + 10)
console.log(addTen(2));   // Output: 12 (10 + 2)
console.log(addTen(7));   // Output: 17 (10 + 7)
```

3)Write a program to create a counter using closures.The center should have 2 functions:-increment & decrement

Answer:

```
function createCounter() {
   let count = 0; // private variable

   function increment() {
      count++;
      console.log(count);
   }

   function decrement() {
      count--;
      console.log(count);
   }

    return { increment, decrement }; // return both functions as an object
}

// Usage
const counter = createCounter();

counter.increment(); // 1
counter.increment(); // 2
counter.decrement(); // 1
```

4)Implement registration process with promise

Answer:

```
function registerUser(username, password) {
    return new Promise((resolve, reject) => {
        // checking if the username already exists ( in db)
        const existingUsers = ['nirma', 'khushboo', 'janani'];

        if (!username || !password) {
            reject('Username and password are required.');
        } else if (existingUsers.includes(username)) {
            reject('Username already exists.');
        } else {
            // registration process (storing data)
            setTimeout(() => {
                resolve('Registration successful!');
            }, 2000); //  async operation (saving to a database)
        }
    });
}

// Usage
registerUser('nirma', 'password123')
    .then((message) => {
        console.log(message); // Username "nirma" already exists
    })
    .catch((error) => {
        console.error(error); // This will print the error message
    });

registerUser('newuser', 'password456')
    .then((message) => {
        console.log(message); // Registration successful!
```

```
  })
  .catch((error) => {
      console.error(error); // This will print the error message if failed
  });

registerUser('khushboo', 'mypassword')
  .then((message) => {
    console.log(message); // Username "khushboo" already exists
  })
  .catch((error) => {
    console.error(error); // This will print the error message
  });
```