JavaScript Fundamentals

Q 1. Lexical Scoping with an Outer and Inner Function

Lexical scoping demonstration:

```
function outerFunction(outerParam) {
    let outerVar = 'Outer variable';

    return function innerFunction() {
        console.log('Outer parameter:', outerParam); // Accessing outerParam
        console.log('Outer variable:', outerVar);     // Accessing outerVar
    };
}

// Example usage:
const inner = outerFunction('Outer argument');
inner(); // Even though outerFunction has finished, innerFunction still has
access to outerParam and outerVar

// Output:
// Outer parameter: Outer argument
// Outer variable: Outer variable
```

Q 2. Basic Usage of Regular Expressions

Function to test a regex pattern against a string:

```
function testRegex(pattern, str) {
    const regex = new RegExp(pattern);
    return regex.test(str);
}

// Example usage:
console.log(testRegex('hello', 'hello world')); // Output: true
console.log(testRegex('\\d+', 'abc123'));       // Output: true
console.log(testRegex('world$', 'hello world')); // Output: true
console.log(testRegex('^world', 'hello world')); // Output: false
```

Q 3. Using Character Classes in Regular Expressions

Function to search for character classes in a string:

```
function searchCharacterClasses(str) {
    const digits = str.match(/\d/g) || [];
    const upperCaseLetters = str.match(/[A-Z]/g) || [];
    const lowerCaseLetters = str.match(/[a-z]/g) || [];
    const specialChars = str.match(/[^a-zA-Z0-9]/g) || [];

    return {
        digits,
        upperCaseLetters,
        lowerCaseLetters,
        specialChars
    };
}

// Example usage:
const result = searchCharacterClasses("Hello World 123! @#");
console.log(result);

// Output:
// {
//   digits: ['1', '2', '3'],
//   upperCaseLetters: ['H', 'W'],
//   lowerCaseLetters: ['e', 'l', 'l', 'o', 'o', 'r', 'l', 'd'],
//   specialChars: [' ', '!', ' ', '@', '#']
// }
```

Q 4. Extracting Specific Parts of a Match Using Regex Groups

Function to extract parts of a date using groups:

```
function extractDateParts(pattern, str) {
    const regex = new RegExp(pattern);
    const match = str.match(regex);

    if (match) {
        return {
            day: match[1],
```

```
            month: match[2],
            year: match[3]
        };
    } else {
        return 'No match found';
    }
}

// Example usage:
const datePattern = /(\d{2})-(\d{2})-(\d{4})/;
const dateStr = "Today's date is 21-09-2024.";
const result = extractDateParts(datePattern, dateStr);
console.log(result);

// Output:
// { day: '21', month: '09', year: '2024' }
```

Q 5. Shipping Application: Calculate Delivery Time Based on Package Type

Program using switch statement:

```
function calculateDeliveryTime(packageType) {
    switch (packageType) {
        case 'standard':
            console.log('Estimated delivery time: 3-5 days');
            break;
        case 'express':
            console.log('Estimated delivery time: 1-2 days');
            break;
        case 'overnight':
            console.log('Estimated delivery time: Next day');
            break;
        default:
            console.log('Invalid package type');
    }
}

// Example usage:
calculateDeliveryTime('standard');  // Output: Estimated delivery time: 3-5
days
calculateDeliveryTime('express');   // Output: Estimated delivery time: 1-2
```

```
days
calculateDeliveryTime('overnight'); // Output: Estimated delivery time: Next
day
calculateDeliveryTime('economy');   // Output: Invalid package type
```