Javascript Fundamentals Assignment Answers

Q1. Write a JavaScript function called outerFunction that takes a parameter and returns an inner function. The inner function should access both the parameter of outerFunction and a variable declared within outer Function. Demonstrate how lexical scoping allows the inner function to maintain access to these variables even after outer Function has finished executing.

Ans:-
```
function outerFunction(parameter) {
   var internalVariable = 10; // Variable declared within outerFunction


   // Inner function defined within outerFunction
   function innerFunction() {
      // Accessing both the parameter of outerFunction and the
internalVariable
      console.log("Parameter of outerFunction:", parameter);
      console.log("Internal variable of outerFunction:",
internalVariable);
   }


   // Returning the inner function
   return innerFunction;
}


// Example usage:
var innerFunc = outerFunction("Hello");
```

innerFunc(); // This will log "Hello" and "10" to the console.

Q2. Create a JavaScript program that demonstrates the basic usage of regular expressions. Write a function that takes a regex pattern and a string as input and returns true if there is a match, and false otherwise. Test the function with various patterns and strings.

Ans:- // Function to test a regex pattern against a string

```
function testRegex(pattern, string) {
   // Creating a regular expression object with the given pattern
   var regex = new RegExp(pattern);

   // Using the test method of the regular expression object to check
for a match
   return regex.test(string);
}

// Testing the function with various patterns and strings
console.log(testRegex("hello", "hello world")); // true
console.log(testRegex("hello", "world"));      // false

console.log(testRegex("\\d", "123"));          // true (checks for any
digit)
console.log(testRegex("\\d", "abc"));          // false

console.log(testRegex("^[A-Z]", "Hello"));      // true (checks if string
starts with an uppercase letter)
```

```
console.log(testRegex("^[A-Z]", "hello"));      // false


console.log(testRegex("a|b", "a"));          // true (checks if string
contains 'a' or 'b')
console.log(testRegex("a|b", "c"));          // false


console.log(testRegex("\\d{3}-\\d{2}-\\d{4}", "123-45-6789")); //
true (checks for a SSN format)
console.log(testRegex("\\d{3}-\\d{2}-\\d{4}", "12-3456-7890")); //
false.
```

Q3. Write a JavaScript program that demonstrates the use of character classes in regular expressions. Create a function that searches for specific character classes in a given string and returns the matches. Test the function with patterns for digits, uppercase letters, lowercase letters, and special characters.

Ans:- function findCharacterClasses(inputString) {

```
  // Define regular expression patterns for character classes
  const digitPattern = /\d/g; // Matches any digit
  const uppercasePattern = /[A-Z]/g; // Matches any uppercase letter
  const lowercasePattern = /[a-z]/g; // Matches any lowercase letter
  const specialCharPattern = /[^a-zA-Z0-9\s]/g; // Matches any
special character (not alphanumeric or whitespace)


  // Find matches for each character class
  const digits = inputString.match(digitPattern) || [];
  const uppercaseLetters = inputString.match(uppercasePattern) || [];
```

```
    const lowercaseLetters = inputString.match(lowercasePattern) || [];
    const specialChars = inputString.match(specialCharPattern) || [];

    // Return the matches
    return {
        digits: digits,
        uppercaseLetters: uppercaseLetters,
        lowercaseLetters: lowercaseLetters,
        specialChars: specialChars
    };
}

// Test the function with a sample string
const testString = "Hello 123 World! @#";

const characterClasses = findCharacterClasses(testString);

console.log("Digits:", characterClasses.digits);
console.log("Uppercase Letters:", characterClasses.uppercaseLetters);
console.log("Lowercase Letters:", characterClasses.lowercaseLetters);
console.log("Special Characters:", characterClasses.specialChars);
```

Q4. Create a JavaScript program that takes a regex pattern and a string as input. Write a function that not only checks if there is a match but also extracts specific parts of the matched text using groups. Test the function with patterns that include groups to

capture different parts of a date (e.g., day, month, and year) from a given string.

Ans:- function extractDateParts(pattern, string) {

```
    // Creating a regular expression object with the given pattern
    var regex = new RegExp(pattern);

    // Executing the regular expression pattern against the string
    var match = regex.exec(string);

    if (match) {
        // Extracting matched groups
        var groups = match.slice(1); // Extracting capture groups
excluding the full match

        return groups;
    } else {
        return null; // Return null if no match is found
    }
}

// Test the function with different date patterns
const dateString = "Today is 2024-02-12";

// Using a pattern to capture year, month, and day separately
const datePattern = /(\d{4})-(\d{2})-(\d{2})/;
const dateParts = extractDateParts(datePattern, dateString);


if (dateParts) {
  console.log("Year:", dateParts[0]);
  console.log("Month:", dateParts[1]);
  console.log("Day:", dateParts[2]);
} else {
  console.log("No date found in the string.");
}
```

Q5. You are building a shipping application. Write a program that takes the type of package ("standard", "express", or "overnight") and uses a switch statement to calculate and print the estimated delivery time based on the package type. For example, "standard" might take 3-5 days, "express" 1-2 days, and "overnight" would be delivered the next day.

Ans:- function calculateDeliveryTime(packageType) {

```
  // Initialize a variable to hold the estimated delivery time
  let estimatedTime;


  // Use a switch statement to calculate the estimated delivery time
based on the package type
  switch (packageType) {
    case "standard":
      estimatedTime = "3-5 days";
      break;
    case "express":
            estimatedTime = "1-2 days";
            break;
        case "overnight":
            estimatedTime = "next day";
            break;
        default:
            estimatedTime = "Unknown"; // Handle unknown package
types
            break;
    }

    // Print the estimated delivery time to the console
    console.log(`Estimated delivery time for ${packageType} package:
${estimatedTime}`);
}

// Test the function with different package types
calculateDeliveryTime("standard");
calculateDeliveryTime("express");
calculateDeliveryTime("overnight");
calculateDeliveryTime("prime"); // Unknown package type.
```

COMPLETE