PW SKILL ASSIGNMENT

Rajkamal Yadav

Question 1:

Create an arrow function called square that takes a number as an argument and returns its square. Use the arrow function to calculate the square of a given number and display the result.

Answer:

```
// Arrow function to calculate the square of a number
const square = (number) => number * number;

// Example usage
const num = 8;
const result = square(num);

console.log(`The square of ${num} is ${result}`);
```

Out Put:

The square of 8 is 64

Question 2:

Create a JavaScript function called generateGreeting that takes a name as an argument and returns a personalized greeting message. Use this function to greet three different people.

Answer:

```
// Function to generate a personalized greeting message
function generateGreeting(name) {
   return `Hello, ${name}! Welcome to our website.`;
}

// Example usage
const person1 = 'Rajkamal Yadav';
const person2 = 'Anil Kumar Pal';
const person3 = 'Pravin Yadav';
```

```
console.log(generateGreeting(person1));
console.log(generateGreeting(person2));
console.log(generateGreeting(person3));
```

Out Put:

Hello, Rajkamal Yadav! Welcome to our website.
Hello, Anil Kumar Pal! Welcome to our website.
Hello, Pravin Yadav! Welcome to our website.

Question 3:

Create an IIFE (Immediately Invoked Function Expression) that calculates the square of a number and immediately displays the result.

Answer:

```
// IIFE to calculate the square of a number and display the result
(function(number) {
    const result = number * number;
    console.log(`The square of ${number} is ${result}`);
}) (10);   // You can change this number to any value you want to square
```

Out Put:

The square of 10 is 100

Question 4:

Write a JavaScript function called calculateTax that takes an income as an argument and returns the amount of tax to be paid. Use a closure to handle different tax rates based on income ranges. Test the function with various incomes.

Answer:

```
// Function to calculate tax based on income ranges using a closure
function calculateTax(income) {
    // Inner function to determine the tax rate based on income
```

```
    function getTaxRate(income) {
        if (income <= 20000) {
            return 0.1; // 10% tax rate for income up to $20,000
        } else if (income <= 50000) {
            return 0.2; // 20% tax rate for income between $20,001 and $50,000
        } else {
            return 0.3; // 30% tax rate for income above $50,000
        }
    }

    // Calculate the tax based on the income and the tax rate
    const taxRate = getTaxRate(income);
    return income * taxRate;
}

// Test the function with various incomes
const income1 = 15000;
const income2 = 35000;
const income3 = 80000;

console.log(`Tax on ${income1} income: ${calculateTax(income1)}`);
console.log(`Tax on ${income2} income: ${calculateTax(income2)}`);
console.log(`Tax on ${income3} income: ${calculateTax(income3)}`);
```

Out Put:

Tax on $15000 income: $1500
Tax on $35000 income: $7000
Tax on $80000 income: $24000

Question 5:

Write a JavaScript function called factorial that calculates the factorial of a non-negative integer using recursion. Test the function with different inputs.

Answer:

```
// Recursive function to calculate the factorial of a non-negative integer
function factorial(n) {
    // Base case: factorial of 0 or 1 is 1
```

```
    if (n === 0 || n === 1) {
        return 1;
    }
    // Recursive case: n * factorial of (n-1)
    return n * factorial(n - 1);
}

// Test the function with different inputs
const num1 = 5;
const num2 = 0;
const num3 = 7;

console.log(`Factorial of ${num1} is ${factorial(num1)}`);
console.log(`Factorial of ${num2} is ${factorial(num2)}`);
console.log(`Factorial of ${num3} is ${factorial(num3)}`);
```

Out Put:

Factorial of 5 is 120
Factorial of 0 is 1
Factorial of 7 is 5040

Question 6:

Write a JavaScript function called curry that takes a function as an argument and returns a curried version of that function. The curried function should accept arguments one at a time and return a new function until all arguments are provided. Then, it should execute the original function with all arguments. Test the curry function with a function that adds two numbers.

Answer:

```
// Function to create a curried version of a given function
function curry(fn) {
    // Recursive helper function to accumulate arguments
    return function curried(...args) {
        if (args.length >= fn.length) {
            // If all expected arguments are provided, call the original function
            return fn(...args);
```

```
    } else {
      // Otherwise, return a new function that expects the next argument
      return function(...nextArgs) {
        return curried(...args, ...nextArgs);
      };
    }
  };
}

// Example function that adds two numbers
function add(a, b) {
  return a + b;
}

// Create a curried version of the add function
const curriedAdd = curry(add);

// Test the curried function
console.log(curriedAdd(2)(3)); // Outputs: 5
console.log(curriedAdd(10)(20)); // Outputs: 30
```

Out Put:

5
30