(Notes) Functions

Question 1: Function Code

Explanation: This code defines a function `square` that calculates the square of a number and a function `displaySquare` that logs the square of 5 using the square function.

Question 2: What is IIFE?

Explanation: IIFE stands for *Immediately Invoked Function Expression*. It is a function that is executed immediately after it is defined. Example 1 shows a simple IIFE, while Example 2 demonstrates an IIFE with nested functions.

Question 3: Closure

Explanation: The code demonstrates closure where the inner function `displayName` has access to the variable `name` defined in the outer function `init`.

Question 4: Function Scope

Explanation: This code showcases function scope and variable shadowing. In Q-1, the `mul` function uses the variables `num1` and `num2` from the global scope. In Q-2, the `setTimeout` function inside a loop creates a closure but uses the global `i` variable.

Question 5: Function Hoisting

Explanation: Function hoisting allows calling a function before it is declared. The code demonstrates hoisting with a function declaration and how it behaves differently from non-hoisted functions.

Question 6: Params vs. Arguments

Explanation: This code defines a function `fn` with parameters `a`, `x`, `y`, and a rest parameter `numbers`, showcasing how parameters are declared and used in functions.

Question 7: Spread operator and rest operator

Explanation: The code shows the usage of the rest parameter in a function declaration and the spread operator to pass an array as individual arguments to the function.

Question 8: Callback

Explanation: This code demonstrates a callback function `greeting` being passed as an argument to the `processUserInput` function, allowing dynamic behavior based on user input.

Question 9: Arrow functions

Explanation: Arrow functions are shown in two forms: one with explicit return and braces, and another with implicit return when the function body is a single expression.

Question 10: this

Explanation: The code defines an object `user` with methods `rc1` and `rc2`. `rc1` uses an arrow function, resulting in `this` referring to the global object, while `rc2` uses a regular function, allowing `this` to refer to the object itself.
