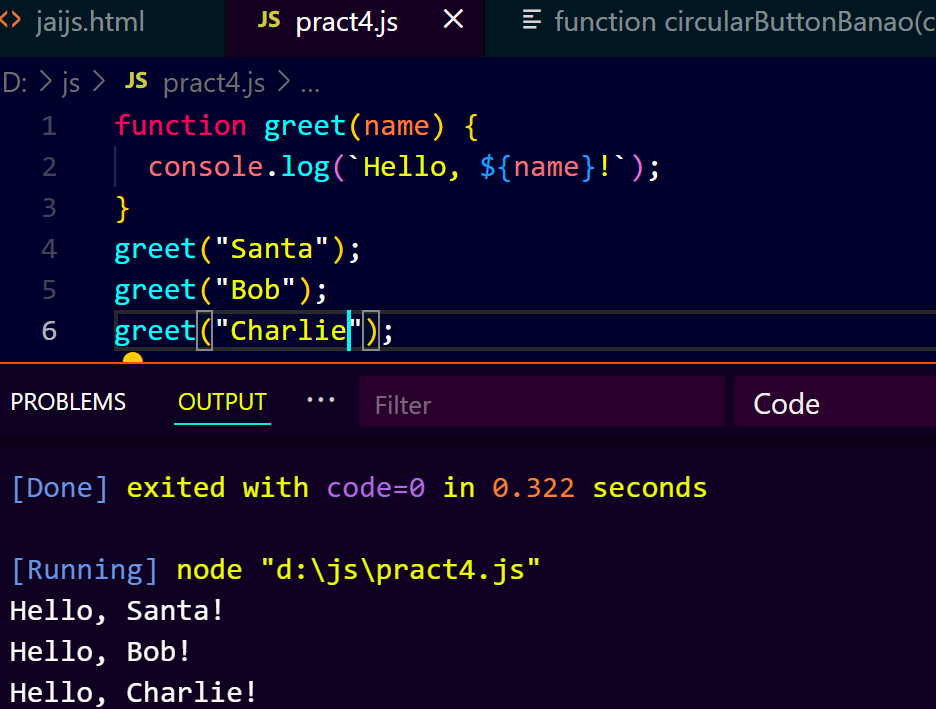
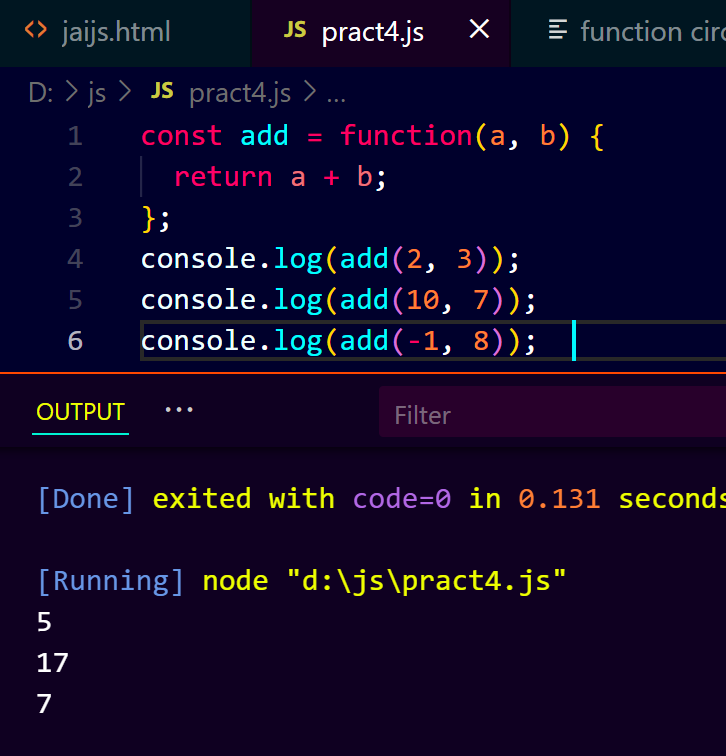
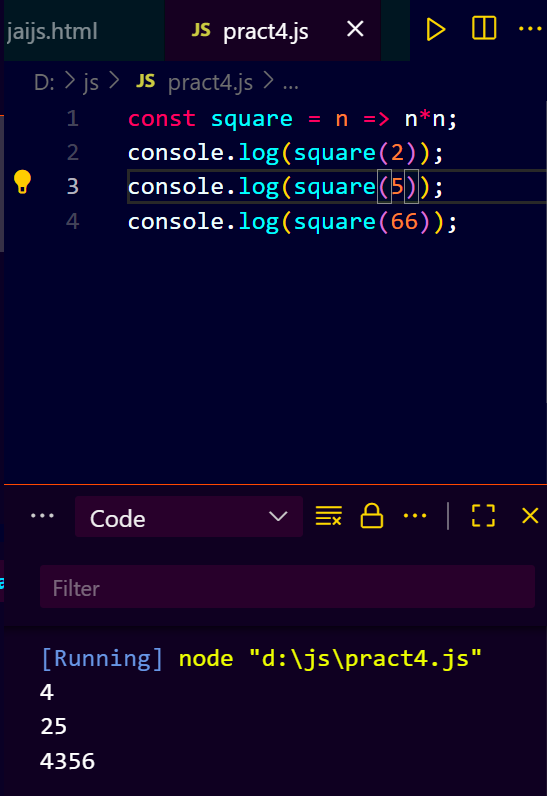
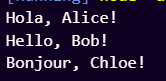
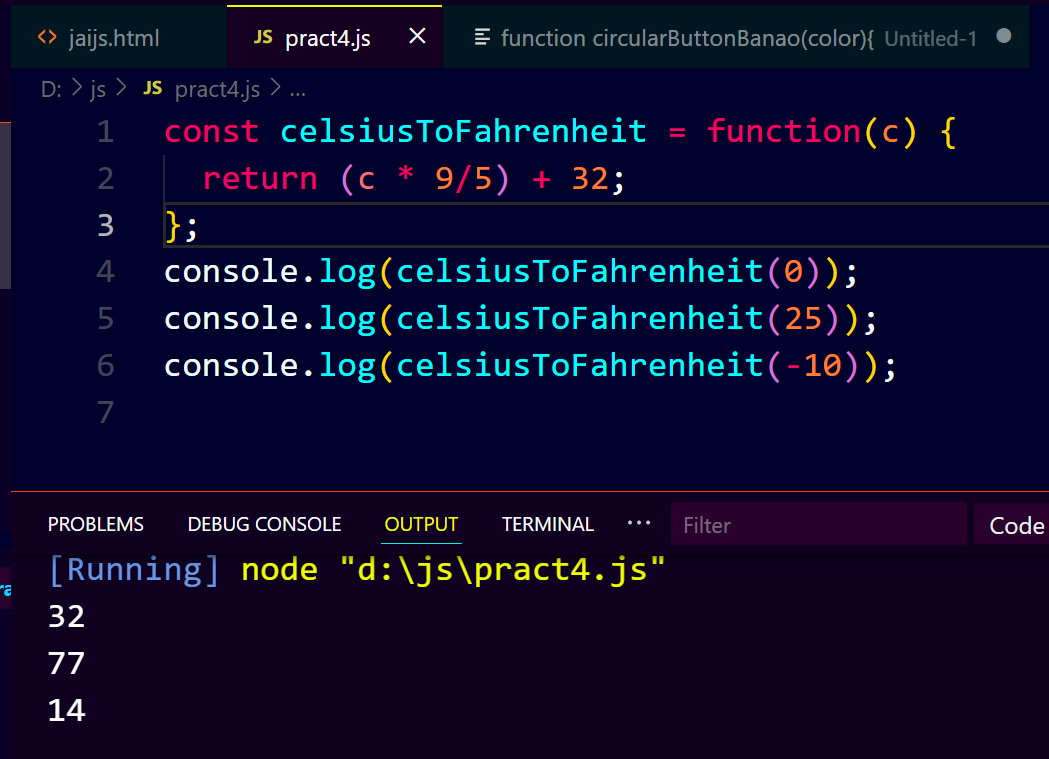
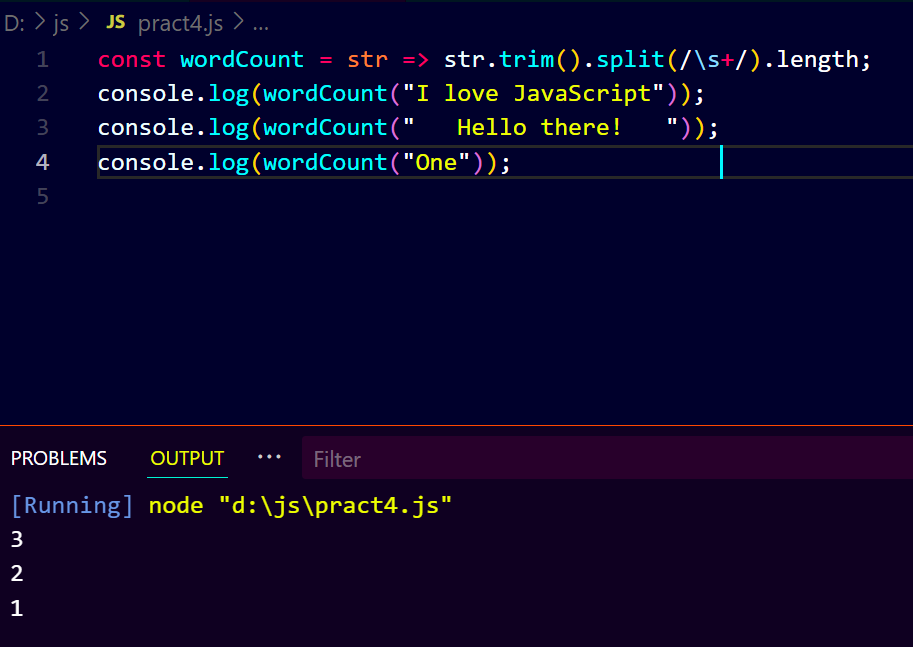
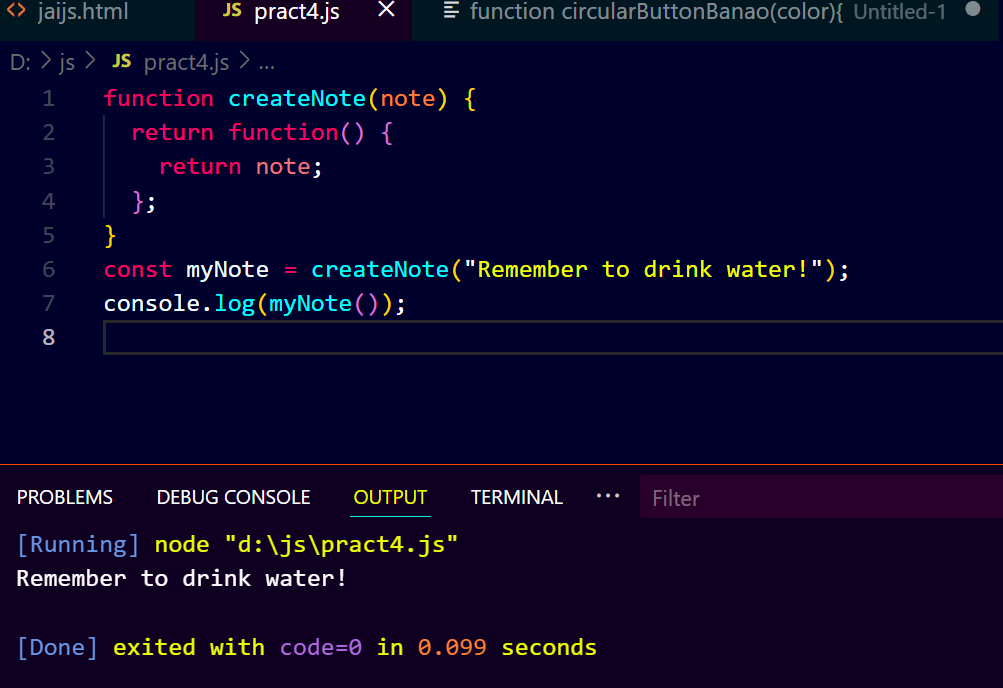
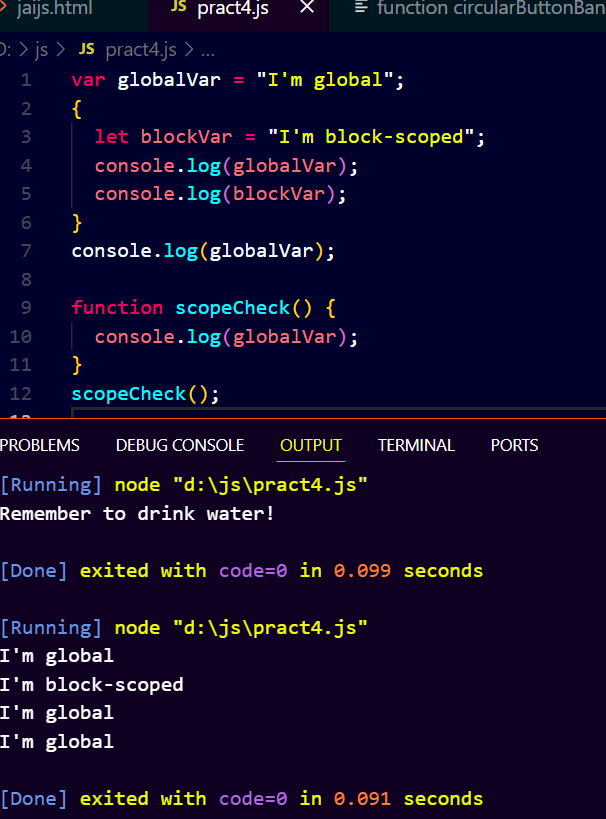
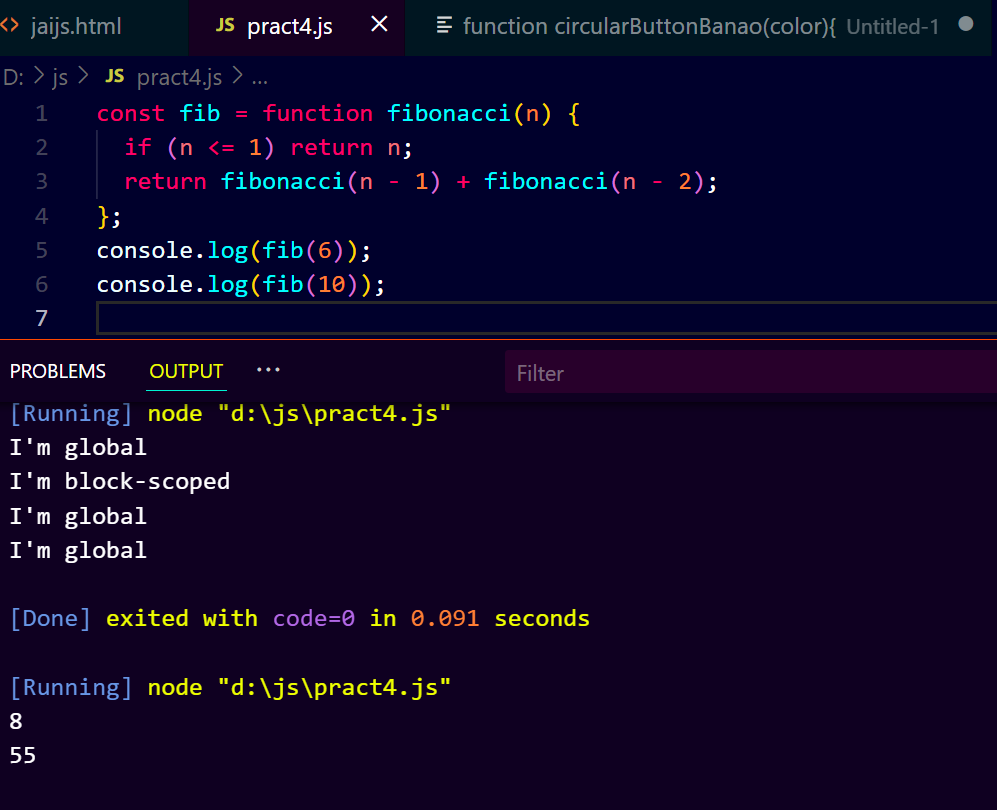
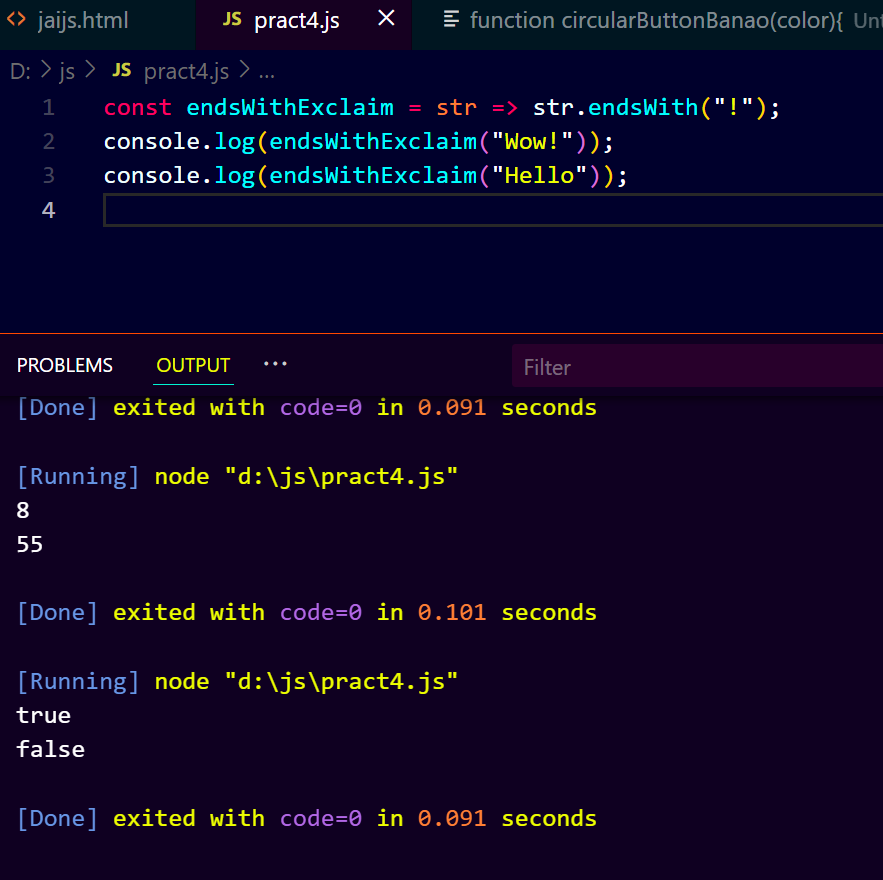
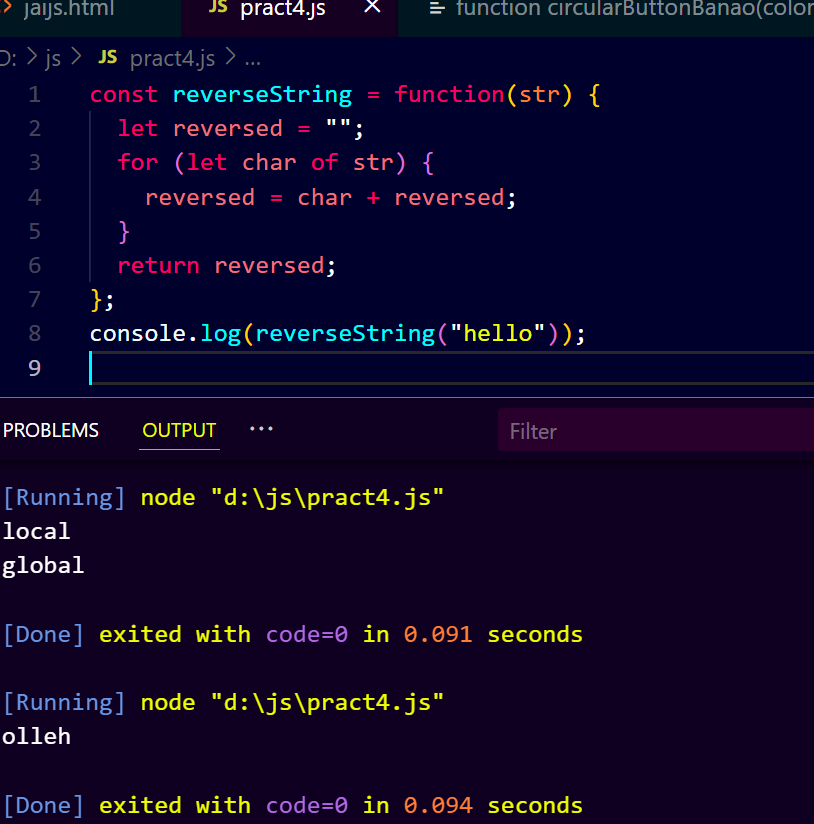
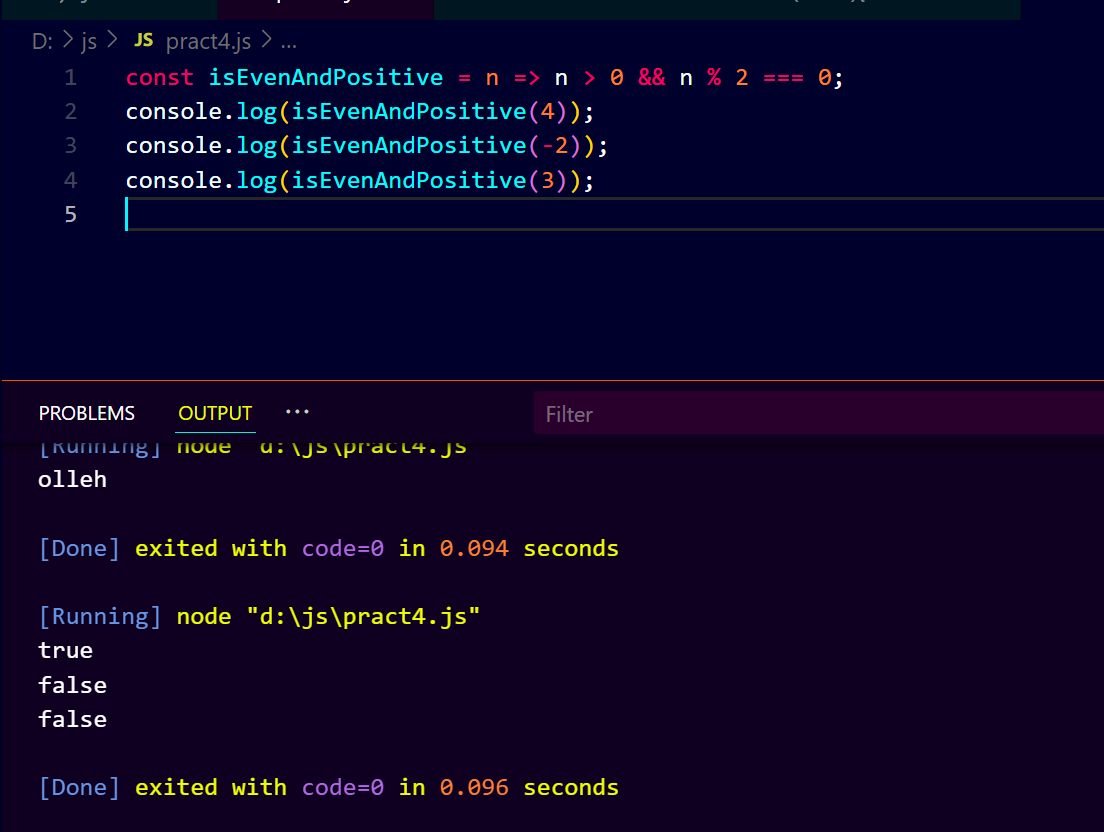
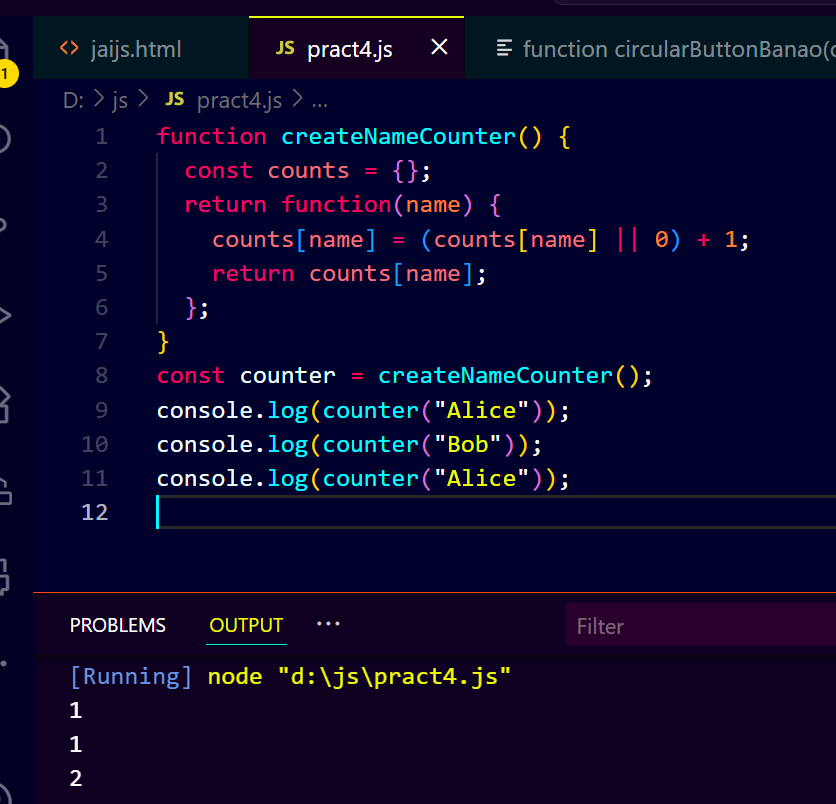
## **Easy Level:**

1. **Function Declaration:** Write a function that takes a name and prints "Hello, <name>!".  
    Call it with different names.
2. **Function Expression:** Create a function expression that takes two numbers and returns their sum.  
    Test it with at least 3 sets of numbers.
3. **Arrow Function:** Write an arrow function that returns the square of a number.  
    Test it with numbers 2, 5, and 10.
4. **Function Declaration with Default Parameter:** Write a function that takes a name and a language (default "English") and prints "Hello" in that language.  
    Example: ("Alice", "Spanish") → "Hola, Alice!".
5. **Function Expression – Simple Math:** Create a function expression that converts Celsius to Fahrenheit.  
    Formula: (C × 9/5) + 32.
6. **Arrow Function – Word Count:** Make an arrow function that takes a string and returns the number of words in it.  
    Example: "I love JavaScript" → 3.
7. **Simple Closure – Personal Note:** Write a closure that stores a note and returns a function to read it later.

## **Moderate Level:**

1. **Scope Practice:** Create a global variable and a block-scoped variable.  
    Demonstrate which ones are accessible inside and outside a function and a block { }.
2. **Function Expression + Recursion:** Write a named function expression that finds the nth Fibonacci number.  
    Test with n = 6 and n = 10.
3. **Arrow Function String Check:** Create an arrow function that checks if a sentence ends with "!".  
    Test it with both true and false cases.
4. **Simple Closure:** Write a function createGreeter that stores a greeting message and returns a function that prints it when called.
5. **Scope Example – Variable Shadowing:** Create a function where a local variable has the same name as a global variable.  
    Show how JavaScript chooses which one to use.
6. **Function Expression – Reverse String:** Make a function expression that reverses a string without using built-in .reverse().
7. **Arrow Function – Multiple Checks:**Write an arrow function that checks if a number is:
   1. Even
   2. Positive  
       Return true only if both conditions are met.  
      
8. **Closure – Personalized Counter:** Create a closure that counts how many times a specific person’s name is entered.  
    Example: If "Alice" is entered three times, it should return 3.



## **Advanced Level**

1. **Closure for Counter:**Create a closure-based counter that allows incrementing, decrementing, and resetting the count.  
    Each operation should be a separate function.
2. **Private Data with Closure:** Make a createBankAccount function that stores a balance.  
    Provide deposit, withdraw, and checkBalance methods.  
    Ensure balance can’t be accessed directly from outside.
3. **Mixed Function Types:**Function Declaration: Generates a random number.

Function Expression: Checks if the number is prime.

Arrow Function: Prints "Prime" or "Not Prime" based on the result.  
 Use all three in sequence.

1. **Scope Chain Debugging:**Write code with three nested functions (outer, middle, inner) each having variables with the same name.  
   Log which one gets accessed in the innermost function and explain why.