JavaScript Contest (2) : You can do it (Think broadly)

Problem Statement (1): “The Time Machine”

In a parallel universe, there exists a time machine that can only travel to different years within the 21st century. The time machine operates on JavaScript fuel, a unique type of fuel that is generated from JavaScript code.

Your task is to write a JavaScript function `timeTravel(year, month, day)` that simulates a journey in this time machine.

Function Input

- `year`: A four-digit integer representing the year (2000 to 2099).
- `month`: An integer (1 to 12) representing the month.
- `day`: An integer (1 to 31) representing the day.

Function Output The function should return a string in the format: “The time machine has travelled to {day}-{month}-{year}”.

Constraints

- The function should validate the input and handle incorrect or impossible dates (e.g., February 30). In such cases, the function should return: “Invalid date. Please try again.”
- The function should not use any external libraries.

Evaluation Submissions will be evaluated based on the correctness of the output, the handling of edge cases, and the simplicity and readability of the code.

Good luck, and may the best JavaScript time traveller win!

Problem Statement(2): "The Number Guesser"

Objective:

Develop a JavaScript application that plays a guessing game with the user, where the computer secretly picks a number and the user tries to guess it within a limited number of attempts.

Input:

- None (the program generates its own secret number).

Output:

- The program should display messages prompting the user for their guess and providing feedback (e.g., "Too high!", "Too low!", "Correct!").
- The final result should display the number of guesses it took the user to find the correct answer.

Constraints:

- Use only basic JavaScript functions, loops, and conditional statements. No advanced data structures or libraries allowed.
- The secret number should be a random integer between 1 and 100 (inclusive).
- The user should be allowed a maximum of 10 guesses.

Bonus Challenge:

- Implement different difficulty levels with varying secret number ranges and guess limits.
- Allow the user to play multiple rounds with different secret numbers.
- Add hints based on the user's guesses to help them solve the puzzle.

Evaluation:

- Functionality (70%): Does the code correctly play the guessing game within the given constraints?
- User experience (20%): Is the game engaging and easy to understand for the user?
- Efficiency (10%): Is the code well-structured and optimized for performance.

Good Luck with the contest!