**Exercise 1: Replacing Parts of a String**

1. Create a variable called greeting with the value "Hello, user!".

2. Use replace to change "user" to "world" and log the result.

3. Modify the same greeting variable to change all instances of "o" to "0" using replaceAll, then log the new string.

**Exercise 2: Trimming Whitespace**

1. Declare a variable called input with the value " JavaScript ".

2. Use trim to remove whitespace from both sides and log the result.

3. Try using trimStart and trimEnd on input separately, logging each result. Observe the differences.

**Exercise 3: Padding Strings**

1. Create a variable id with the value "42".

2. Use padStart to make id 5 characters long, adding "0" at the start.

3. Use padEnd to make the string 10 characters long by adding "\*".

4. Log each result to observe the padding.

**Exercise 4: Escaping Characters**

1. Create a string variable quote with the following text: She said, "It's time to code!".

2. Use escape characters to properly format the string so that it displays double and single quotes as shown.

3. Add a new line between "She said," and "It's time to code!" using \n.

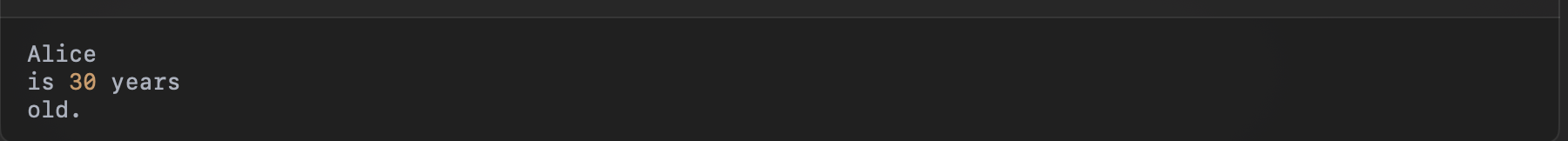
4. Log quote to see the formatted output.

**Exercise 5: Template Literals**

1. Declare two variables: name with "Alice" and age with 30.

2. Using a template literal, create a sentence that says "Alice is 30 years old.", embedding the variables.

3. Modify the sentence to create a multi-line string:



4. Log the result.

**Exercise 6: Identifying Error Types**

1. Write code that triggers the following errors one at a time:

• A SyntaxError by adding a ( without closing it.

• A ReferenceError by trying to log a variable that hasn’t been declared.

• A TypeError by calling a non-function like a function (e.g., 5()).

2. Run your code and note the error messages for each.

**Exercise 7: Handling Errors with try…catch**

1. Write a try...catch block where the try section attempts to parse JSON from the string "{ invalid JSON }".

2. Log an error message inside the catch block, indicating that there was an issue parsing the JSON.

**Exercise 8: Using finally**

1. Modify the try...catch block from Exercise 7 to include a finally block.

2. Inside the finally block, log "Parsing attempt completed."

3. Run your code to confirm the finally block executes regardless of the result.

**Exercise 9: Throwing Custom Errors**

1. Write a function divide that takes two arguments, a and b.

2. Inside divide, throw a custom error with the message "Cannot divide by zero!" if b is zero.

3. Use try...catch to handle this error when calling divide(10, 0).

4. Log a friendly message inside catch if an error is thrown, e.g., "Error: Cannot divide by zero!"