**ASSESSMENT**

**1. Differentiation of let, const, and var in JavaScript.**

* LET- has a block scope, meaning its accessible only within the block its declared in. You can re-assign a let variable, but you can’t re-declare it in the same scope. Use let when you need to re-assign a variable within its scope.
* CONST- it has a block scope like let, however, const variable are immutable, meaning you can’t re-assign them after they’re initialized. Use const whenever possible, as it helps prevent accidental re-assignment and makes your code more predictable.
* VAR- it is the oldest way to declare variables in JavaScript. Unlike let and const, var has a function scope, meaning it’s accessible within the function it’s declared in. You can re-assign a var variable, and it can be declared multiple times in the same scope. Avoid using var in modern JavaScript code.

**2. Concept of Falsy Values in JavaScript.**

* In JavaScript, falsy values are those that evaluate to false when used in a conditional statement. These values are special because they represent the absence of a meaningful value or a state of ‘nothingness’. While they might look like regular data, they behave differently in logical contexts.

In conditional statements, falsy values trigger the ‘false’ branch of the condition. For instance, if (0) or if (“ “) would evaluate to false, this behavior is crucial for controlling program flow and making decisions based on the presence or absence of meaningful data.

* Example of falsy values.

1. False: it is the most obvious falsy value. Its like saying ‘no’ or ‘not true’.
2. 0 (zero): considered falsy because it means ‘nothing’ or ‘empty’.
3. “ “ (empty string): falsy because it has no text inside. Its like empty box with nothing in it.
4. Null: means ‘nothing’ or ‘empty’ on purpose, it’s like saying “I don’t have a value right now.”
5. Undefined: means a variable doesn’t have a value yet. Its like a box that hasn’t been filled with anything.