**Assignment 1**

**Question: 1 Analyse the Behaviour of Named Function in Expression**

A named function in expression means you create a function, give it a name inside, and store it in a variable (box). The name works only inside the function.

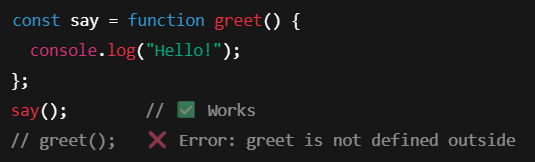
**Behaviour:**

We can **call the function using the variable name**.

The internal function name helps with **debugging or recursion**.

The inner name is **not available outside** the function.

**Code:**

****

* greet() works **only inside** the function. Outside, use say().

**Output:**



**Question 2: Event Listener – Why It Is Slow? GEC Analysis & Closure Solution**

An **event listener** listens for clicks, inputs, or scrolls. It runs a function when that event happens.

**Why It’s Slow:**

Adding **too many listeners** (e.g., to each button) fills up memory.

They stay in **GEC (Global Execution Context)** longer.

Leads to **slow pages** and **memory leaks**.

**GEC (Global Execution Context) Analysis:**

All functions run inside the GEC.

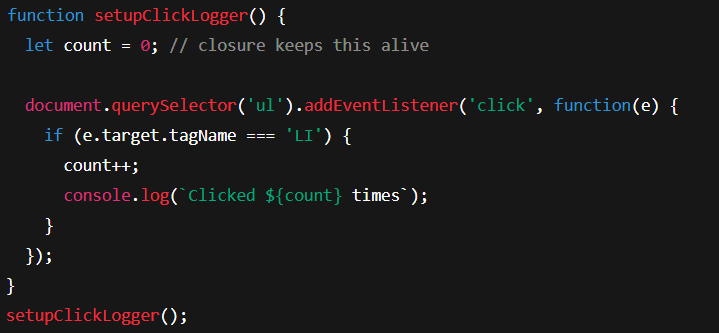
If we keep event listeners active everywhere, memory stays **occupied**.

Garbage Collector **can’t clean them** unless we remove them or manage them properly.

**How to Fix It (with Closure):**

We use **closures** and **event delegation** — attach one listener to the parent and handle all children efficiently.

**Code:**

****

Use **one event listener** + **closure** to avoid slowdowns and save memory.

**Output:**



Use **one event listener** + **closure** to avoid slowdowns and save memory.

**Question 3: Garbage Collector**

The **Garbage Collector** is a memory cleaner. It removes data that is no longer used in the program.

**Behaviour:**

If no one is using an object → it is marked as "garbage".

The garbage collector **automatically deletes** it from memory.

Helps keep your app **fast and light**.

**Code:**

****

**Output:**



Don’t keep unused data or event listeners — let GC clean it.