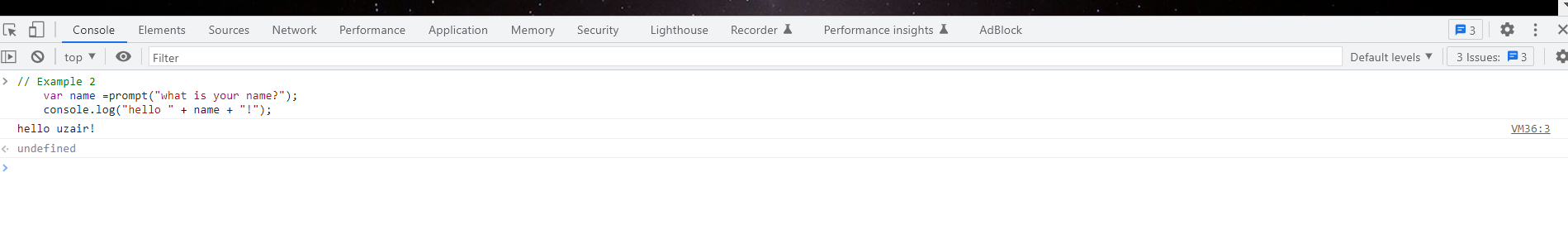
**// What is Execution context:**

The execution context is a place where your javascript code is actually executed and call stack is the collection of your execution context. Which works in the LIFO(Last In First Out).

// Example 01

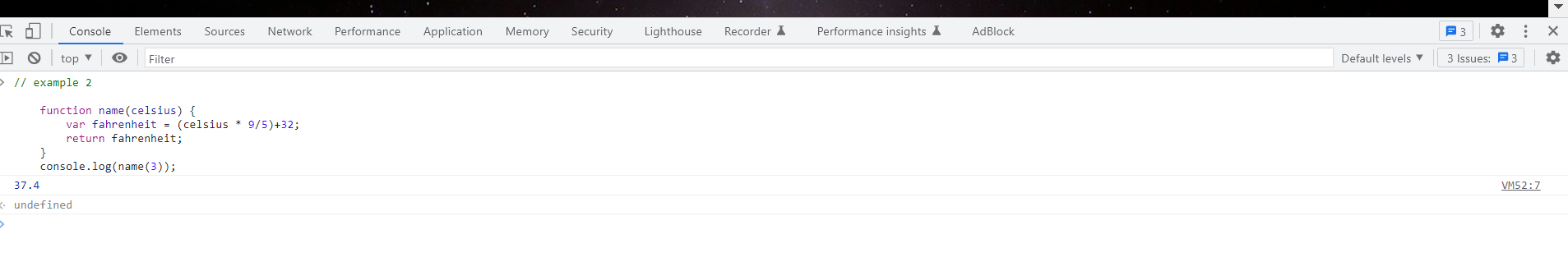
****

Description:

This “prompt()” method is used to javascript code that asks the user to input the name store the value of function.

For example if the user entered “uzair” when prompted for their name the output to the console would be “Hello Uzair!”.

// Example 2



Description:

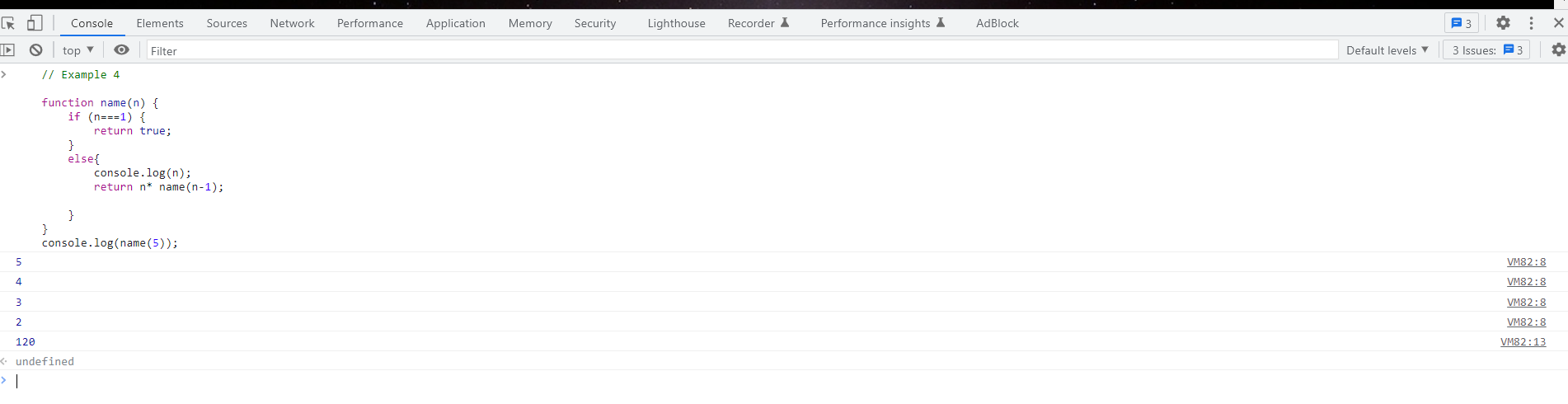
The Celsius temperature value is converted to Fahrenheit by the formula.

(multiplying by 9/5) + 32 Fahrenheit

The function with an argument of 3 represents a tempreture value in Celsius.

They will be return (37.4) is printed.

// Example 3



Description:

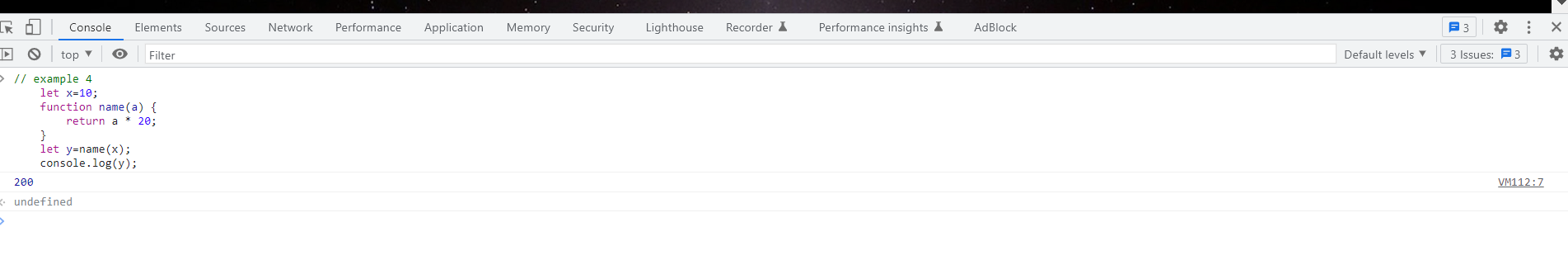
This example of recursive function in javascript. The function factorial a parameter n.

The function calculate the factorial of the given number by multiplying.

If we call the factorial function with an argument 5 first check n is equal to 1 then return true otherwise false.

The output of the function 120. Which is product of function 45321

// Example 4



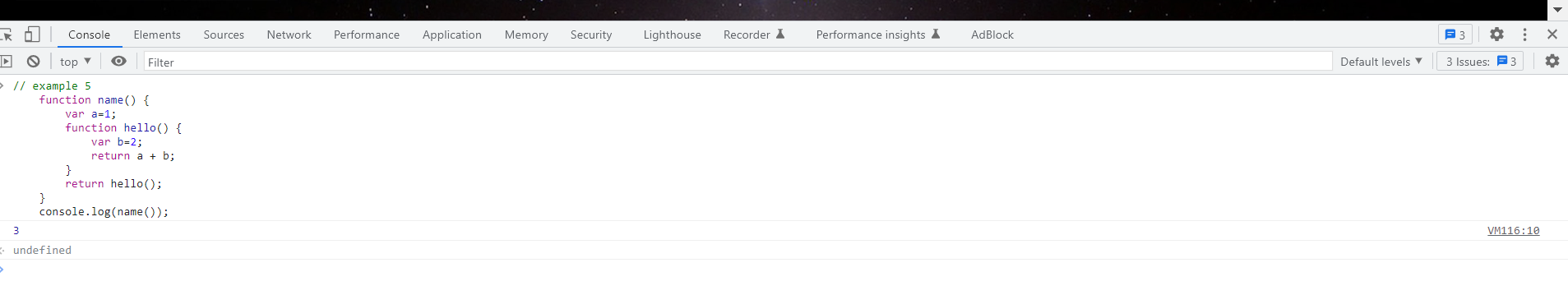
Description:

This example a function a return a value that is stored in a variable. Function name(a) is variable is initialized with a value of 10. Then a function name is declear with parameter a.

Next the function is called with the argument “x”.

The output of the function 200 is stored in a variable y.

// Example 5

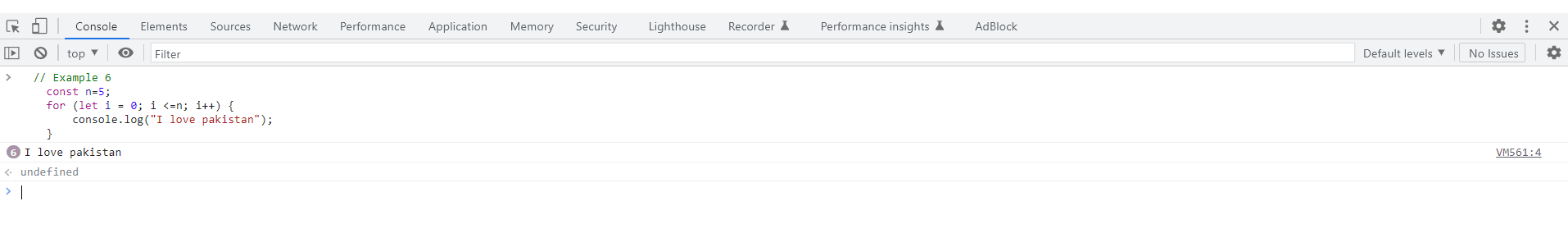


Description:

This method is used to nested function execution context inside the “name” function “hello” is declared. When the hello function is called a new nested function execution context is created.

This execution context also has its own memory space to store variable and function declaration . The output of the function 3. They used to nested function.

// Example 6.



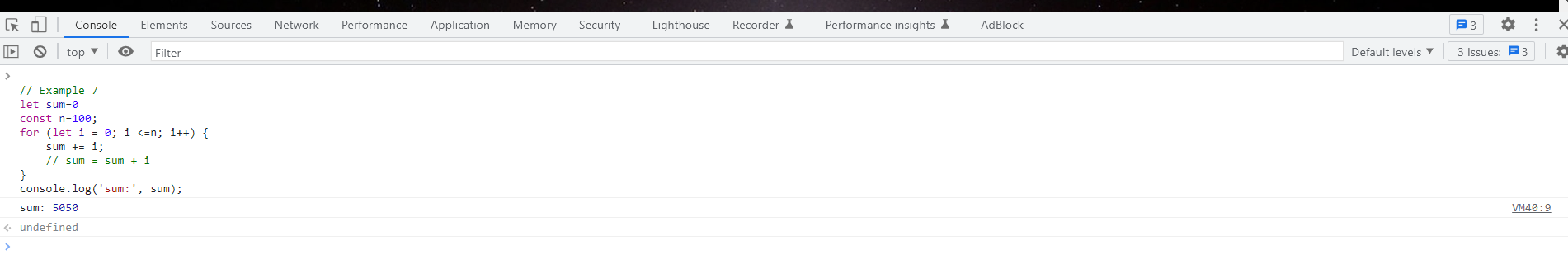
Description:

n is the number of 5. i is block scope of the function. Let i = undefined.

Script: 5; i is the index of number 5. They will block scope to execute to increment of the number.

Then i is the number of 0

// Example 7



// Description

A variable is store to sum =0; they n is undefined other variable store the value of 100.

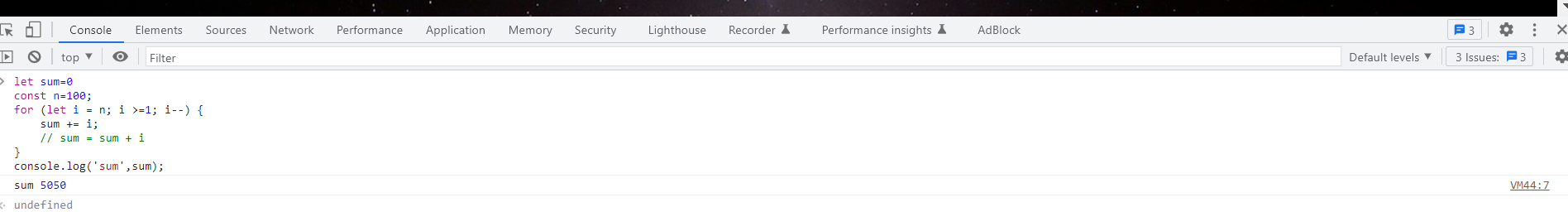
They n = 100 is global scope. They loop is executing context is block scope of the function.

The loop execute then increment of the number of n.

They should i is index of less than or equal to n number of variable.

The console to “sum” is sum multiply by number of n. They will return to multiply by each value of variable.

// Example 8



Description

A variable is store to sum =0; they n is undefined other variable store the value of 100.

They n = 100 is global scope. They loop is executing context is block scope of the function.

They should i is start to n . The i >=1 then decrement of i .

The console to “sum” is sum multiply by number of n. They will return to multiply by each value of variable.

An execution context is an abstract concept of environment where code is javascript is execute and evaluated. Whenever any code is run in javascript it is run inside an execution context.

// Create a global object window

// Create a this object binding which points to global object above.

// set memory space for function and variables.

// store the function declaration in memory heaps and variable decleartion with undefined value in global execution context.

**There are Two Phase of Execution Context:**

|  |  |
| --- | --- |
| **Creation Phase** | **Execution Phase** |
| Global Object: Window | Global Object: Window |
| This: Global Object | This: Global Object |
| x: undefined | x:10 |
| Product: function{…}. | Product: function{…}. |
| Result: undefined | Result: product |