TRAINING REPORT (TR-102)

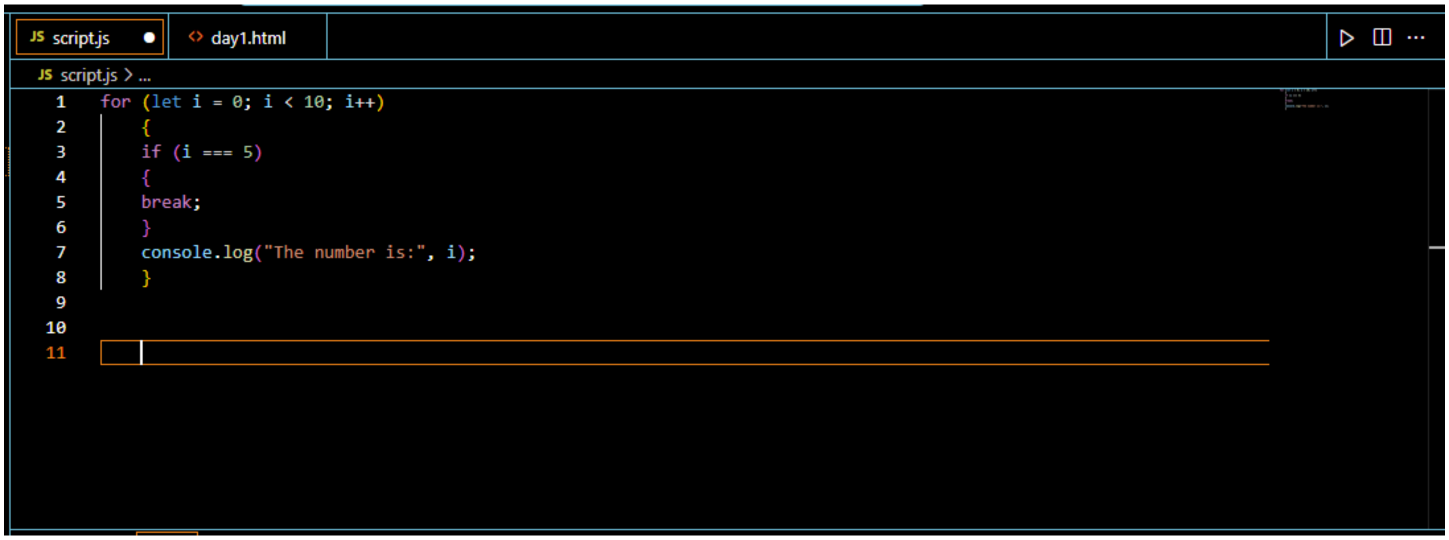
DAY: -17 (1 JULY 2024)

<u>JavaScript Break and Continue: -</u>

The **<u>break statement</u>** "jumps out" of a loop.

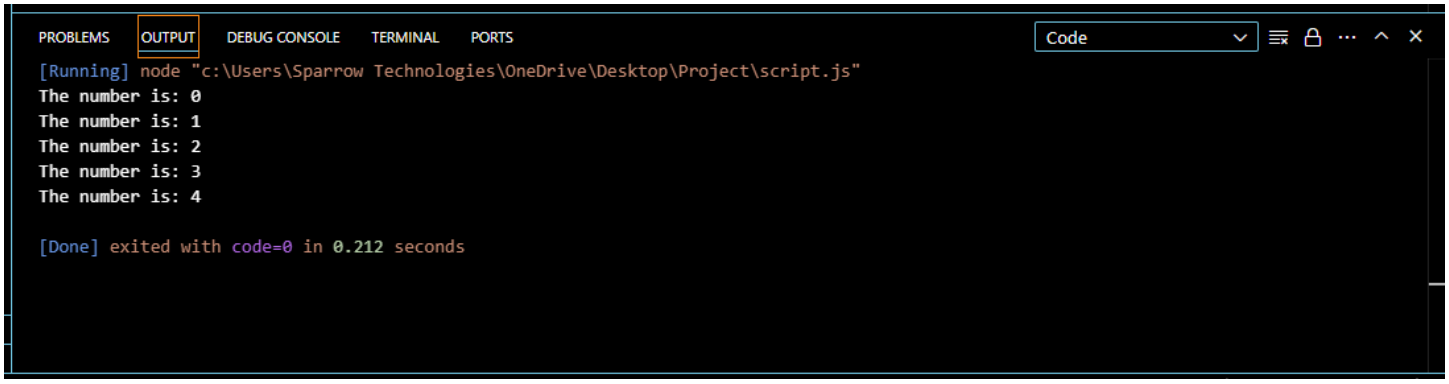
The **<u>continue statement</u>** "jumps over" one iteration in the loop.

<u>The Break Statement: -</u> You have already seen the break statement used in an earlier chapter of thistutorial. It was used to "jump out" of a switch () statement.



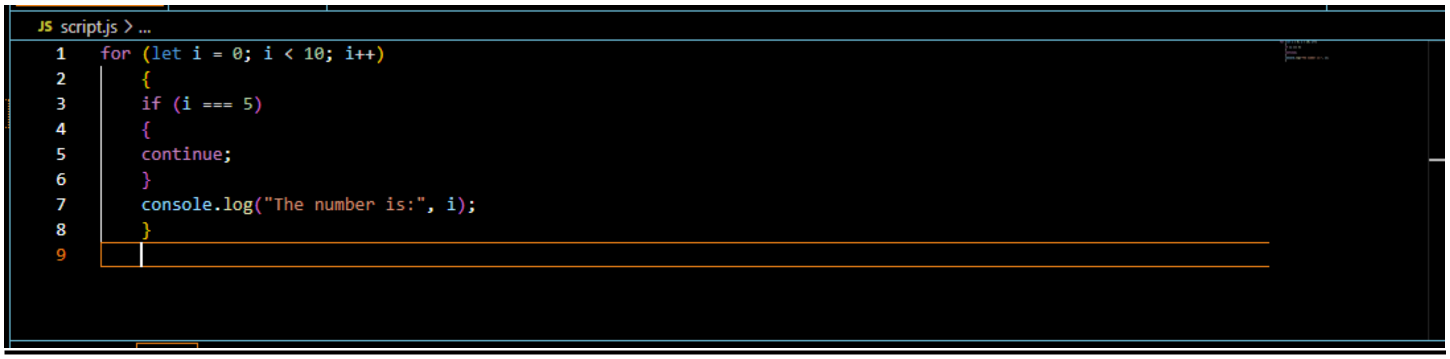
```
for (let i = 0; i < 10; i++)
    {
    if (i === 5)
    {
    break;
    }
    console.log("The number is:", i);
    }
```

<u>OUTPUT: -</u>



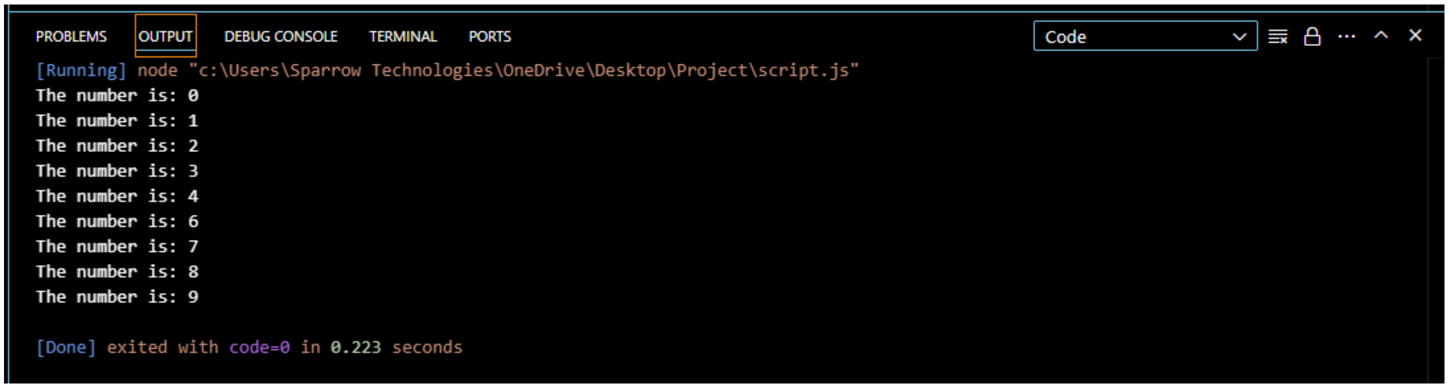
```
[Running] node "c:\Users\Sparrow Technologies\OneDrive\Desktop\Project\script.js"
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4

[Done] exited with code=0 in 0.212 seconds
```

<u>The Continue Statement: -</u> The continue statement breaks one iteration (in the loop), if a specified conditionoccurs, and continues with the next iteration in the loop.



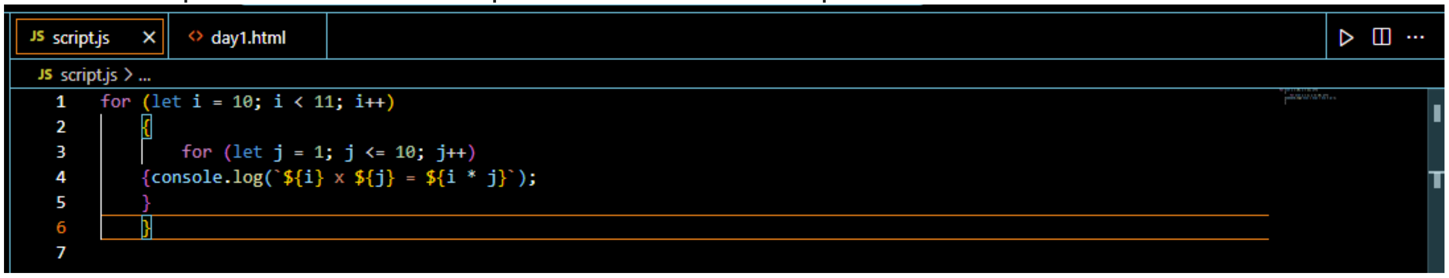
```
for (let i = 0; i < 10; i++)
    {
    if (i === 5)
    {
    continue;
    }
    console.log("The number is:", i);
    }
```

OUTPUT: -



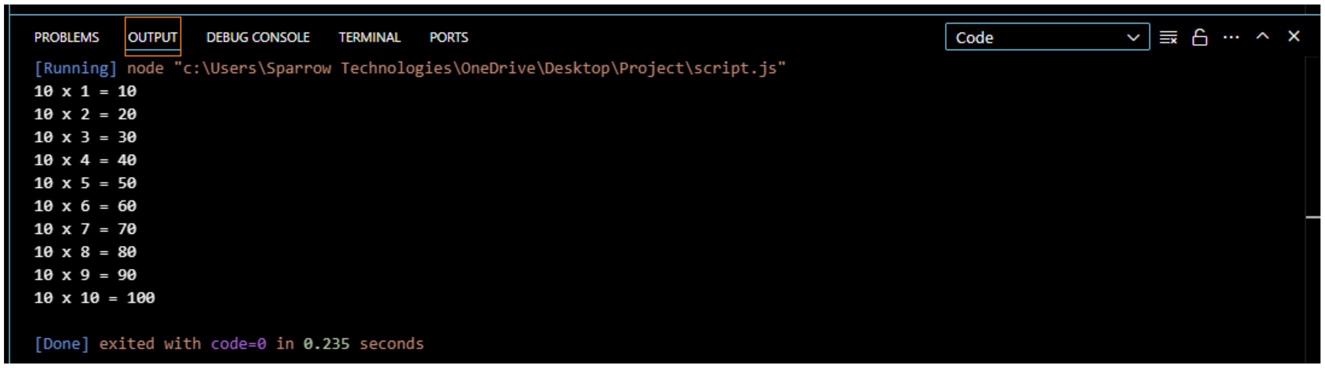
```
[Running] node "c:\Users\Sparrow Technologies\OneDrive\Desktop\Project\script.js"
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 6
The number is: 7
The number is: 8
The number is: 9

[Done] exited with code=0 in 0.223 seconds
```

Nested Loops JavaScript: - A composition of loops is called a nested loop. The most common type of nested loops will be one loop inside another. The first loop is usually called the outer loopwhile the second loop is called the inner loop.



```
for (let i = 10; i < 11; i++)
    {
        for (let j = 1; j <= 10; j++)
    {console.log(`${i} x ${j} = ${i * j}`);
    }
    }
```

OUTPUT: -



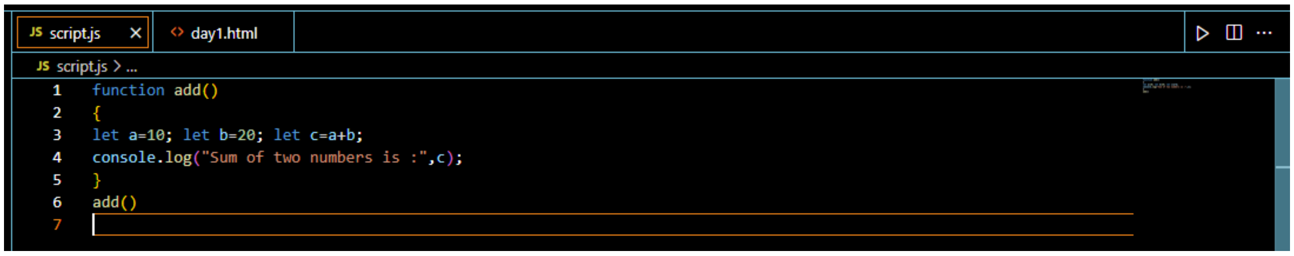
```
[Running] node "c:\Users\Sparrow Technologies\OneDrive\Desktop\Project\script.js"
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100

[Done] exited with code=0 in 0.235 seconds
```

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

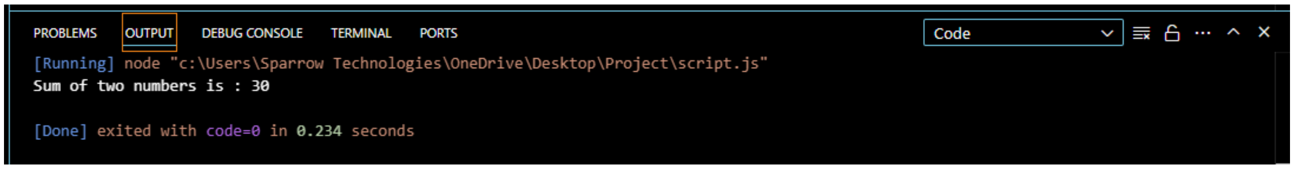
JavaScript Function Syntax: -

- A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().
- Function names can contain letters, digits, underscores, and dollar signs (samerules as variables).
- The parentheses may include parameter names separated by commas: (*parameter1, parameter2, ...*)
- The code to be executed, by the function, is placed inside curly brackets: {}
- Function parameters are listed inside the parentheses () in the function definition.
- Function **arguments** are the **values** received by the function when it is invoked.
- Inside the function, the arguments (the parameters) behave as local variables.



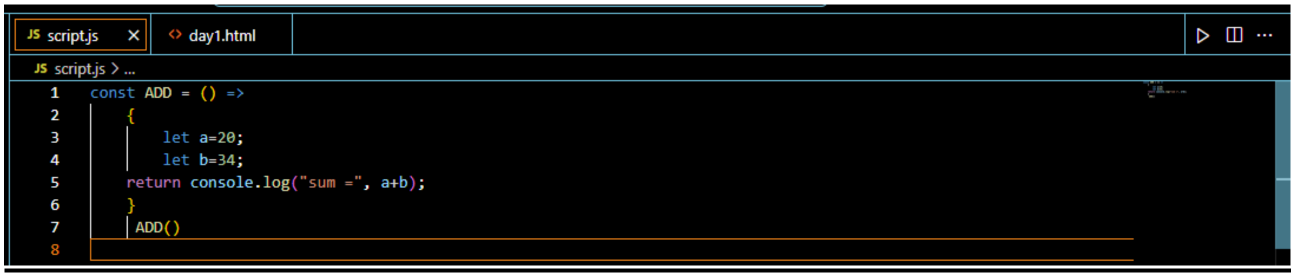
```
function add()
{
let a=10; let b=20; let c=a+b;
console.log("Sum of two numbers is :",c);
}
add()
```

OUTPUT: -



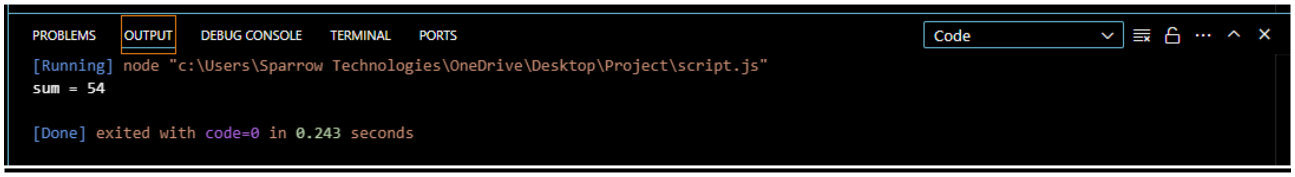
```
[Running] node "c:\Users\Sparrow Technologies\OneDrive\Desktop\Project\script.js"
Sum of two numbers is : 30

[Done] exited with code=0 in 0.234 seconds
```

Arrow function: -



```
const ADD = () =>
    {
        let a=20;
        let b=34;
    return console.log("sum =", a+b);
    }
    ADD()
```

OUTPUT: -



```
[Running] node "c:\Users\Sparrow Technologies\OneDrive\Desktop\Project\script.js"
sum = 54

[Done] exited with code=0 in 0.243 seconds
```

Anonymous Function: - It is a function that does not have any name associated with it. Normally we use the *function* keyword before the function name to define a function in JavaScript,however, in anonymous functions in JavaScript, we use only the *function* keyword without the function name.

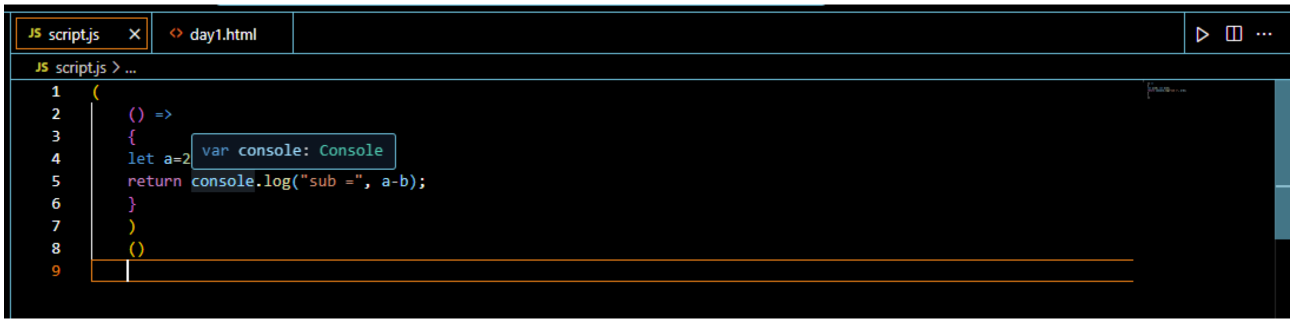
An anonymous function is not accessible after its initial creation, it can only beaccessed by a variable it is stored in as a *function as a value*. An anonymous function can also have multiple arguments, but only one expression.

Syntax: - The below-enlightened syntax illustrates the declaration of an anonymous function using the normal declaration:

```
Function () {
   // Function Body
 }
```

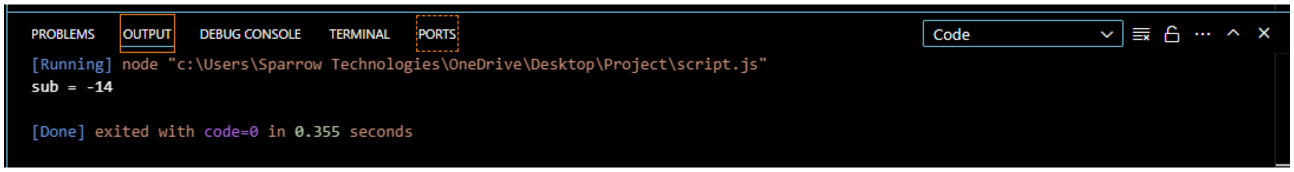
We may also declare an anonymous function using the arrow function technique which is shown below:

```
( () => {
   // Function Body...
} )();
```



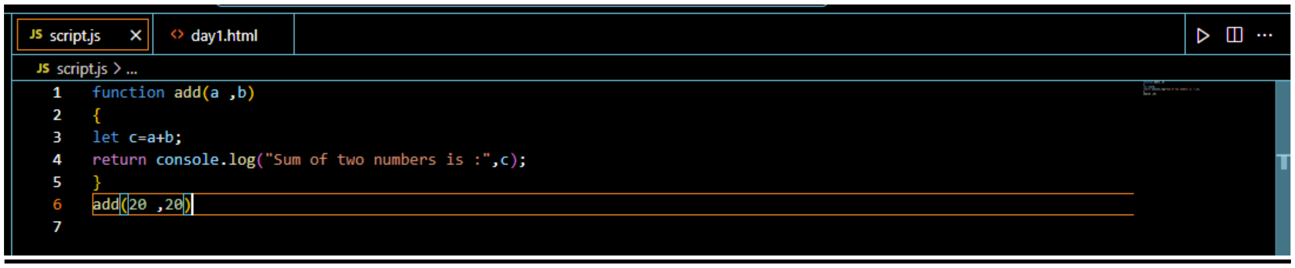
```
(
    () =>
    {
    let a=2 var console: Console
    return console.log("sub =", a-b);
    }
    )
    ()

```

OUTPUT: -



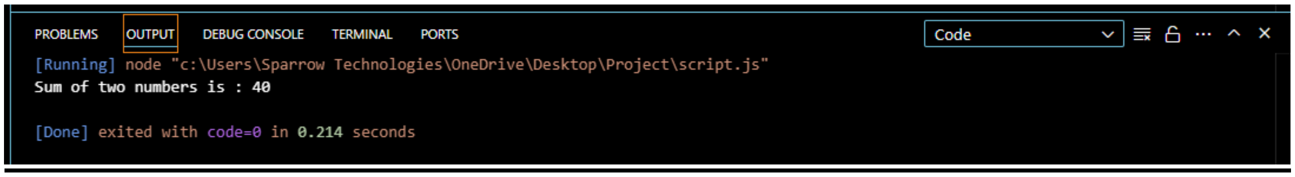
```
[Running] node "c:\Users\Sparrow Technologies\OneDrive\Desktop\Project\script.js"
sub = -14

[Done] exited with code=0 in 0.355 seconds
```

Parameters: -



```
function add(a ,b)
{
let c=a+b;
return console.log("Sum of two numbers is :",c);
}
add(20 ,20)

```

OUTPUT: -



```
[Running] node "c:\Users\Sparrow Technologies\OneDrive\Desktop\Project\script.js"
Sum of two numbers is : 40

[Done] exited with code=0 in 0.214 seconds
```