

LAB ASSIGNMENT NO – 04

Aim: Write a program in Javascript to study conditional Statements, Loops and Functions. Write a Javascript code to change background color of web page automatically after every 5 seconds.

Theory: JavaScript (Js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document.

➤ Conditional Statements - Conditional statements are used to perform different actions based on different conditions.

1. If - The if statement is to specify a block of JavaScript code to be executed if a condition is true.

```
if (condition) {  
    // block of code to be executed if the condition is true  
}
```

2. Else - The else statement is to specify a block of code to be executed if the condition is false.

```
if (condition) {  
    // block of code to be executed if the condition is true  
} else {  
    // block of code to be executed if the condition is false  
}
```

3. Else if - The else if statement is to specify a new condition if the first condition is false.

```
if (condition1) {  
    // block of code to be executed if condition1 is true  
} else if (condition2) {  
    // block of code to be executed if the condition1 is false and  
    condition2 is true  
} else {  
    // block of code to be executed if the condition1 is false and  
    condition2 is false  
}
```

➤ Loops - Loops can execute a block of code a number of times. Loops are handy, if you want to run the same code over and over again, each time with a different value.

1. for - Executes the loop block for a specified number of times under a termination condition.
Syntax - for(Initialization; Terminate Condition; Increment/Decrement)
2. for...in - Executes the loop block through an object's properties.
Syntax - for(variable_name in object) {
 ...
}
3. for...of - Executes the loop block to iterates the iterable instead of object literals.
Syntax - for(variable_name of object) {
 ...
}
4. for...Each – It is a method that calls a function for each element in an array.
5. while - Executes the loop block for a specified number of times under a termination condition.
Syntax - while (terminator condition) {
 ...
}
6. do...while - Similar to the while loop but executes the loop first and evaluates.
Syntax - do {
 ...
}
while (terminator condition);

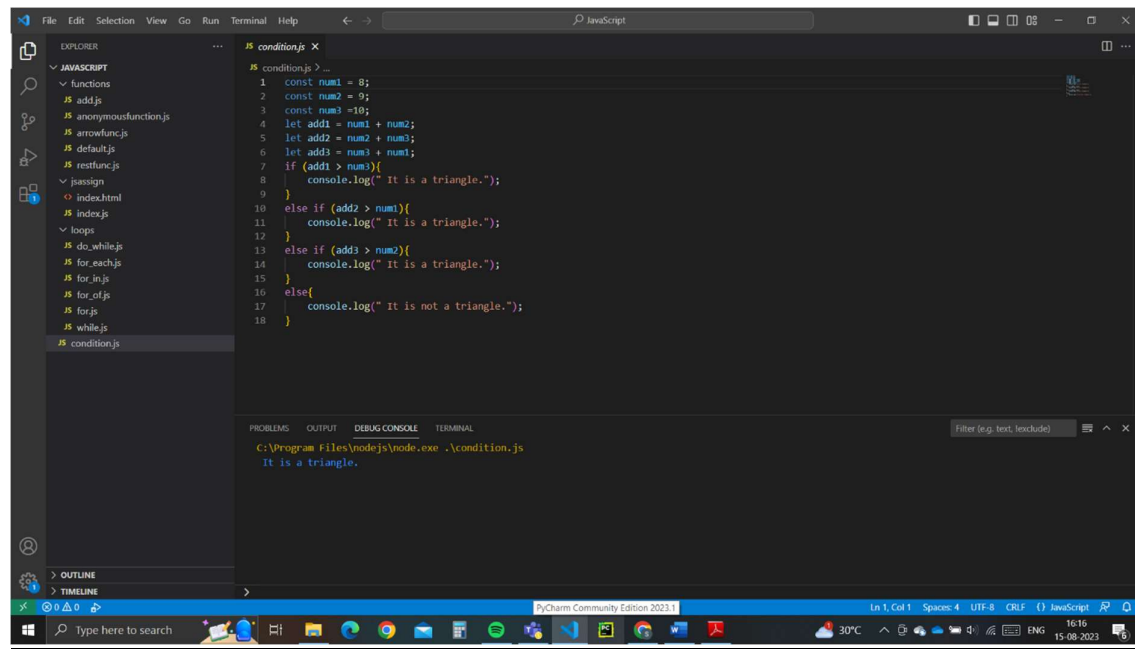
➤ Functions - A JavaScript function is a block of code designed to perform a particular task. A JavaScript function is executed when "something" invokes it (calls it).

1. Returning - Functions may also return the value along with control, back to the caller. Such functions are called as returning functions. A returning function must end with a return statement.
2. Parametrized - It is also possible to provide expressions as default values. Parameters are a mechanism to pass values to functions. Parameters form a part of the function's signature. The parameter values are passed to the function during its invocation.
Syntax : function sum(x = y, y = 1) {
 console.log(x + y);
}

3. Rest - Rest parameters are similar to variable arguments in Java. Rest parameters doesn't restrict the number of values that you can pass to a function. To declare a rest parameter, the parameter name is prefixed with three periods, known as the spread operator.
Syntax : `let func = function(...args) {
 console.log(args);
}`
4. Anonymous Function - Functions that are not bound to an identifier (function name) are called as anonymous functions. These functions are dynamically declared at runtime, functions can accept inputs and return outputs, just as standard functions do.
Syntax : `var res = function([arguments]) { ... }
var f = function() {
 return "hello";
}
console.log(f())`
5. Lambda Function/ Arrow Function - Lambda functions are a concise mechanism to represent anonymous functions. These functions are also called as Arrow functions. There are 3 parts to a Lambda function –
 - Parameters– A function may optionally have parameters.
 - The fat arrow notation/lambda notation(=>): It is also called as the goes to operator.
 - Statements– Represents the function's instruction set.Syntax : `([param1, param2,...param n])=>statement;`

Conclusion : The experiment on JavaScript loops, conditions, and functions provided valuable insights into the fundamental building blocks of programming. Through systematic exploration and practical implementation, we gained a deeper understanding of how loops enable efficient repetitive tasks, how conditions facilitate decision-making processes, and how functions enhance code modularity and reusability.

Code and Output :

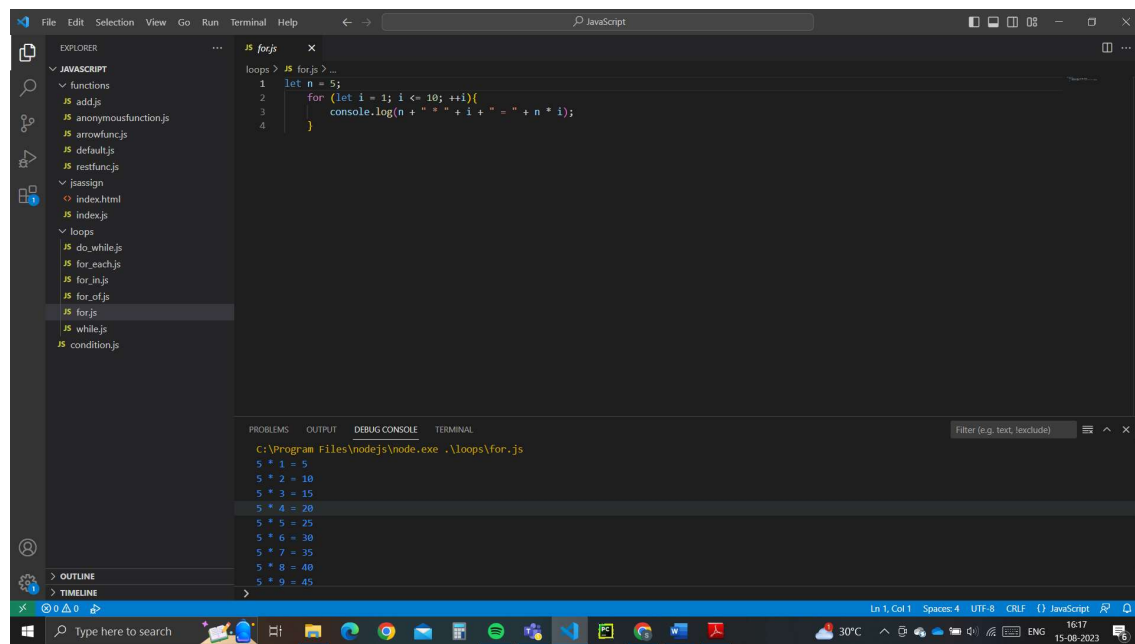


The screenshot shows the VS Code editor with a file named `condition.js` open. The code defines three constants: `num1 = 8`, `num2 = 9`, and `num3 = 10`. It then calculates `add1 = num1 + num2`, `add2 = num2 + num3`, and `add3 = num3 + num1`. The code uses `if` and `else if` statements to check if `add1 > num3`, `add2 > num1`, or `add3 > num2`. If any of these conditions are true, it logs "It is a triangle." to the console. Otherwise, it logs "It is not a triangle." The terminal output shows "It is a triangle."

```
condition.js > ...
1 const num1 = 8;
2 const num2 = 9;
3 const num3 = 10;
4 let add1 = num1 + num2;
5 let add2 = num2 + num3;
6 let add3 = num3 + num1;
7 if (add1 > num3){
8   console.log(" It is a triangle.");
9 }
10 else if (add2 > num1){
11   console.log(" It is a triangle.");
12 }
13 else if (add3 > num2){
14   console.log(" It is a triangle.");
15 }
16 else{
17   console.log(" It is not a triangle.");
18 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

C:\Program Files\nodejs\node.exe .\condition.js
It is a triangle.



The screenshot shows the VS Code editor with a file named `for.js` open. The code defines a constant `n = 5` and uses a `for` loop to iterate from `i = 1` to `i = 10`. Inside the loop, it logs the multiplication of `n` and `i` to the console. The terminal output shows the results of the multiplications: `5 * 1 = 5`, `5 * 2 = 10`, `5 * 3 = 15`, `5 * 4 = 20`, `5 * 5 = 25`, `5 * 6 = 30`, `5 * 7 = 35`, `5 * 8 = 40`, and `5 * 9 = 45`.

```
loops > for.js > ...
1 let n = 5;
2 for (let i = 1; i <= 10; ++i){
3   console.log(n + " * " + i + " = " + n * i);
4 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

C:\Program Files\nodejs\node.exe .\loops\for.js
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45

```
loops > # for_of.js > ...
1 const fruits = ["apple", "grapes", "mango"];
2
3 for (const fruit of fruits) {
4   console.log(fruit);
5 }
6
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

C:\Program Files\nodejs\node.exe .\loops\for_of.js

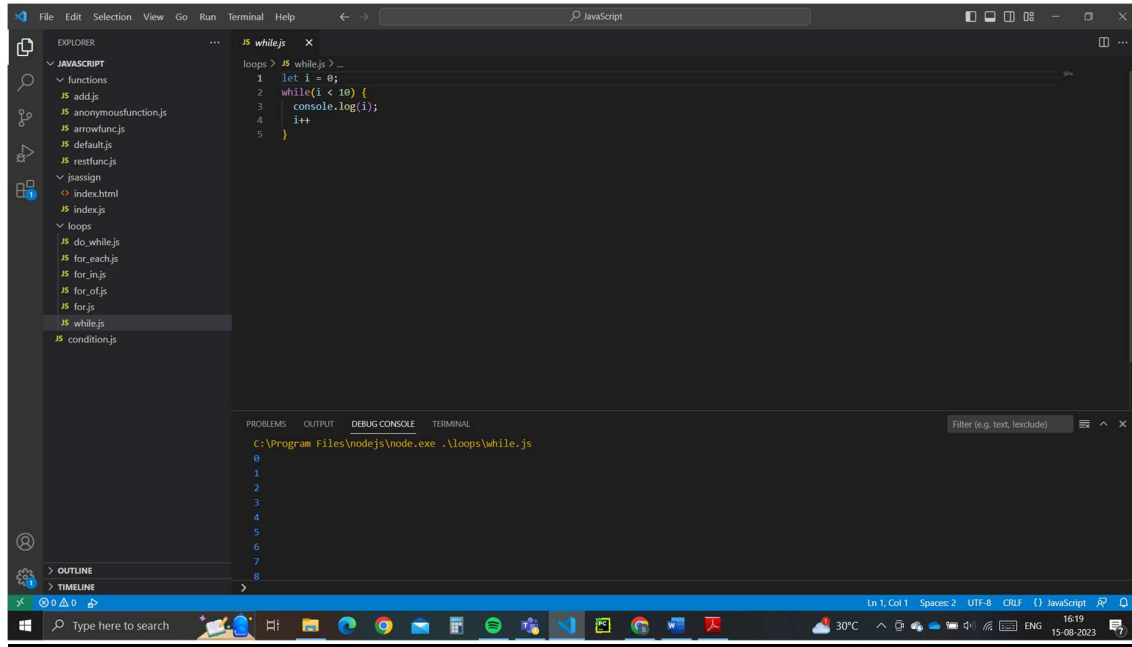
apple
grapes
mango

```
loops > # for_each.js > ...
1 let num = [10,20,30,40,50];
2 num.forEach(num => {
3   console.log(num);
4 });
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

C:\Program Files\nodejs\node.exe .\loops\for_each.js

10
20
30
40
50

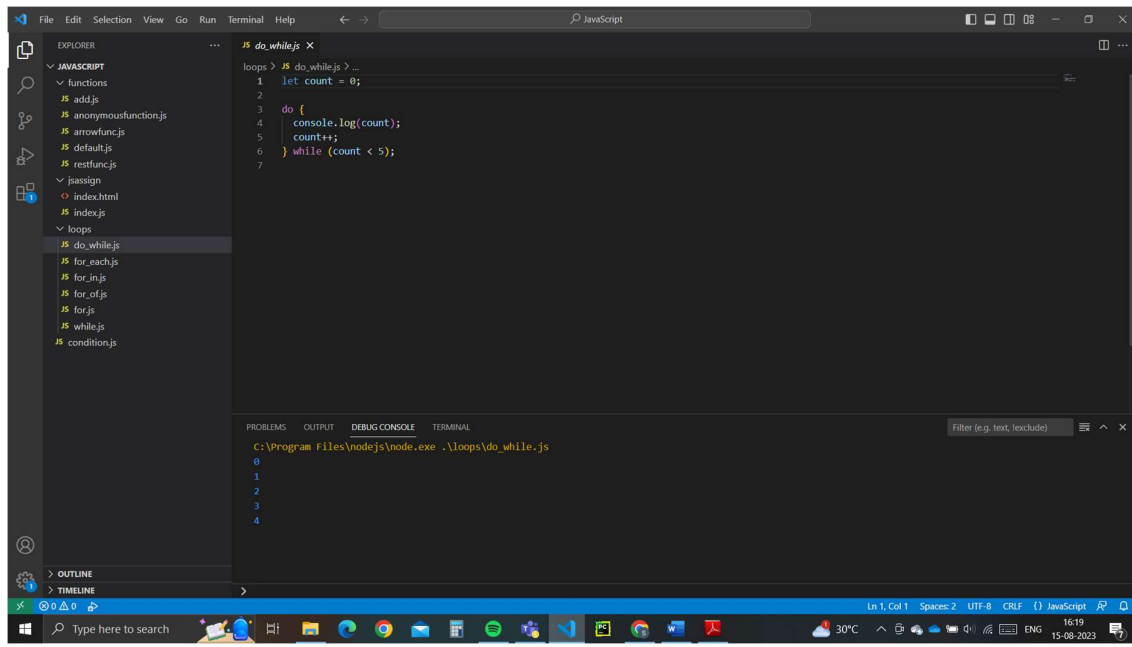


The screenshot shows the Visual Studio Code editor with a JavaScript file named `while.js` open. The file contains a `while` loop that initializes `i` to 0 and logs the value of `i` to the console as long as `i` is less than 10. The terminal output shows the loop running from 0 to 9.

```
loops > while.js ...
1 let i = 0;
2 while(i < 10) {
3   console.log(i);
4   i++;
5 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
C:\Program Files\nodejs\node.exe .\loops\while.js
0
1
2
3
4
5
6
7
8
9
```

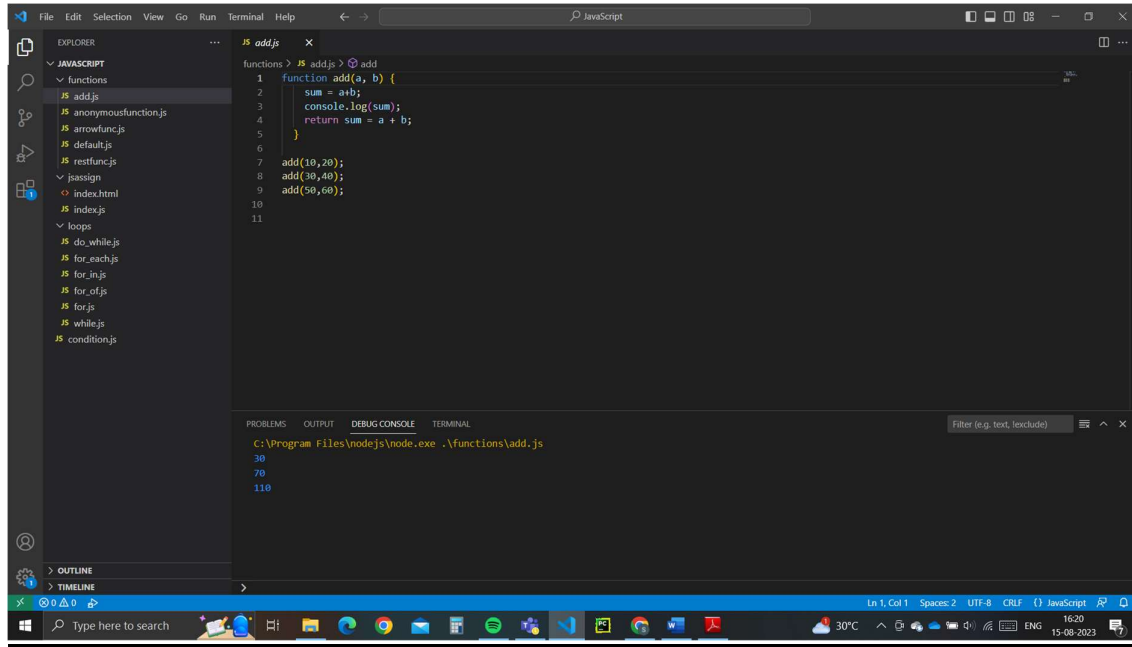


The screenshot shows the Visual Studio Code editor with a JavaScript file named `do_while.js` open. The file contains a `do-while` loop that initializes `count` to 0 and logs the value of `count` to the console as long as `count` is less than 5. The terminal output shows the loop running from 0 to 4.

```
loops > do_while.js ...
1 let count = 0;
2
3 do {
4   console.log(count);
5   count++;
6 } while (count < 5);
7
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
C:\Program Files\nodejs\node.exe .\loops\do_while.js
0
1
2
3
4
```

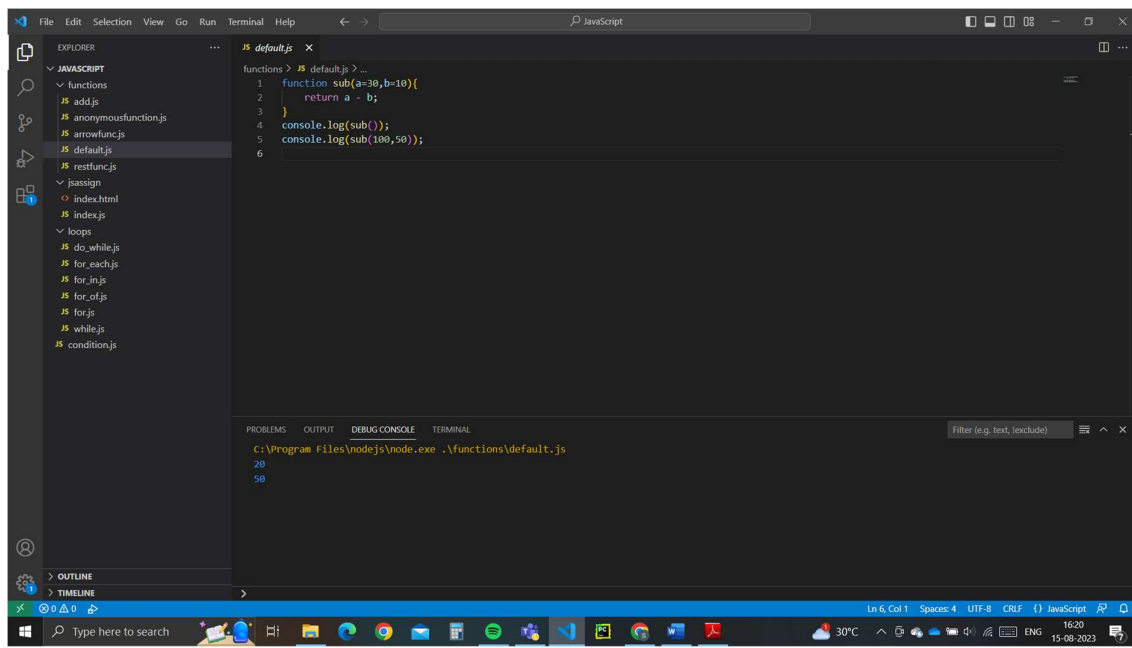


The screenshot shows the Visual Studio Code editor with a JavaScript file named `add.js` open. The file contains a function `add(a, b)` that calculates the sum of two numbers and logs it to the console. The function is called with `add(10, 20)`, `add(30, 40)`, and `add(50, 60)`. The terminal output shows the results: 30, 70, and 110.

```
function add(a, b) {  
  sum = a + b;  
  console.log(sum);  
  return sum = a + b;  
}  
  
add(10, 20);  
add(30, 40);  
add(50, 60);
```

Terminal Output:

```
C:\Program Files\nodejs\node.exe .\functions\add.js  
30  
70  
110
```

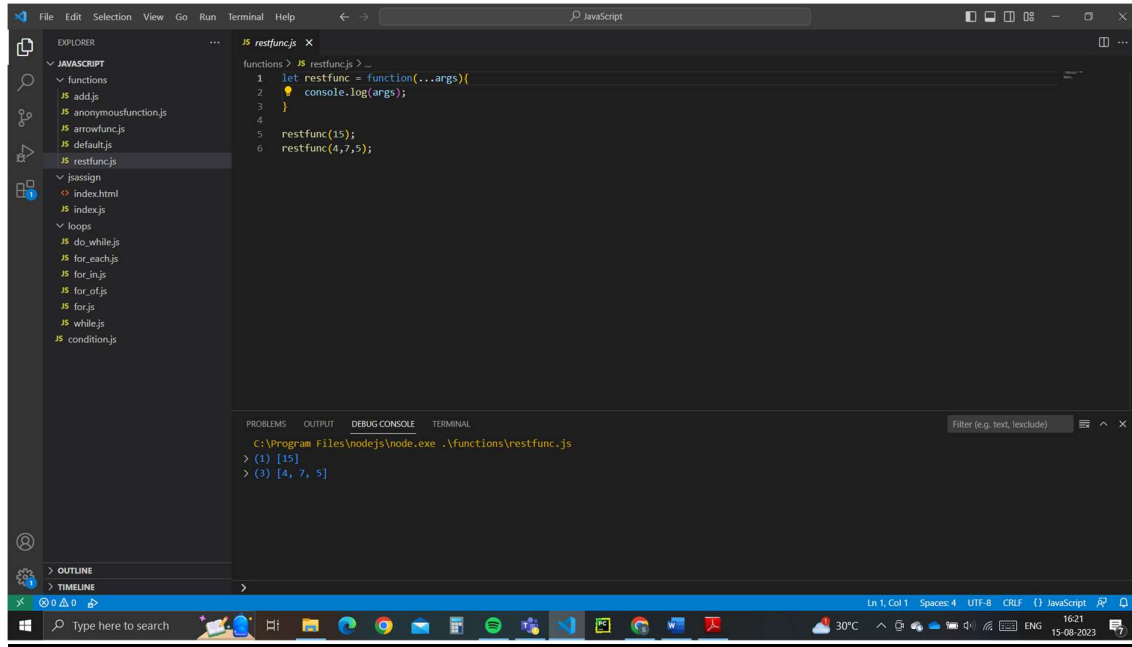


The screenshot shows the Visual Studio Code editor with a JavaScript file named `default.js` open. The file contains a function `sub(a, b)` that calculates the difference of two numbers and logs it to the console. The function is called with `sub()` and `sub(100, 50)`. The terminal output shows the results: 20 and 50.

```
function sub(a, b){  
  return a - b;  
}  
  
console.log(sub());  
console.log(sub(100, 50));
```

Terminal Output:

```
C:\Program Files\nodejs\node.exe .\functions\default.js  
20  
50
```

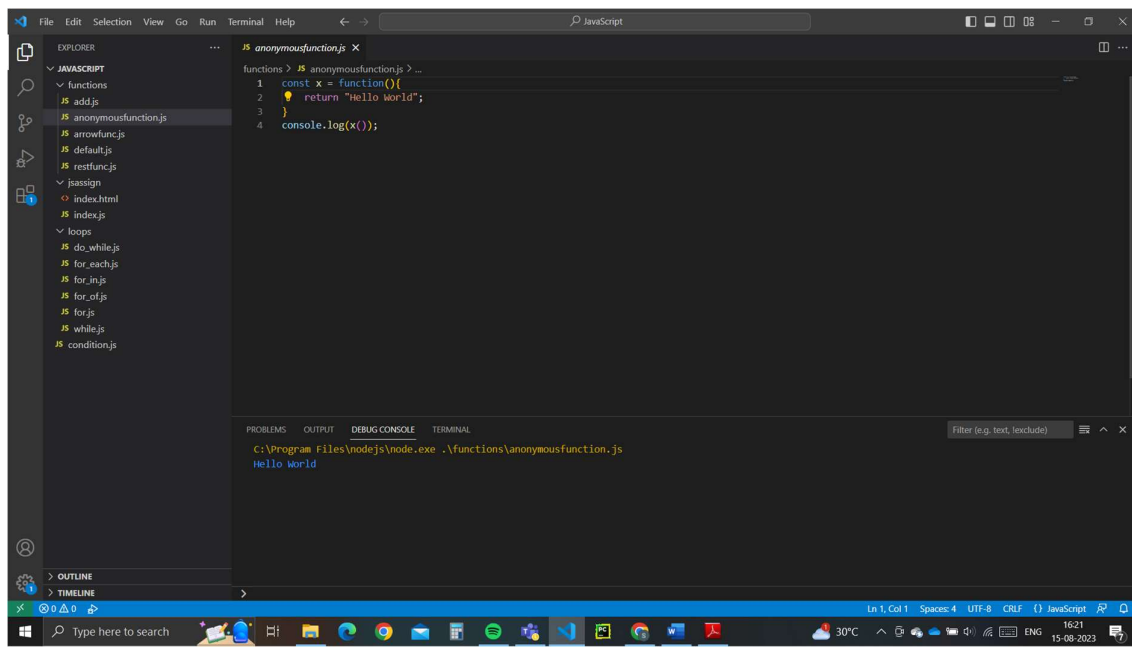


The screenshot shows the Visual Studio Code editor with a JavaScript file named `restfunc.js` open. The code defines a rest function and calls it with arguments. The terminal output shows the function execution results.

```
1 let restfunc = function(...args){
2   console.log(args);
3 }
4
5 restfunc(15);
6 restfunc(4,7,5);
```

Terminal Output:

```
C:\Program Files\nodejs\node.exe .\functions\restfunc.js
> (1) [15]
> (3) [4, 7, 5]
```

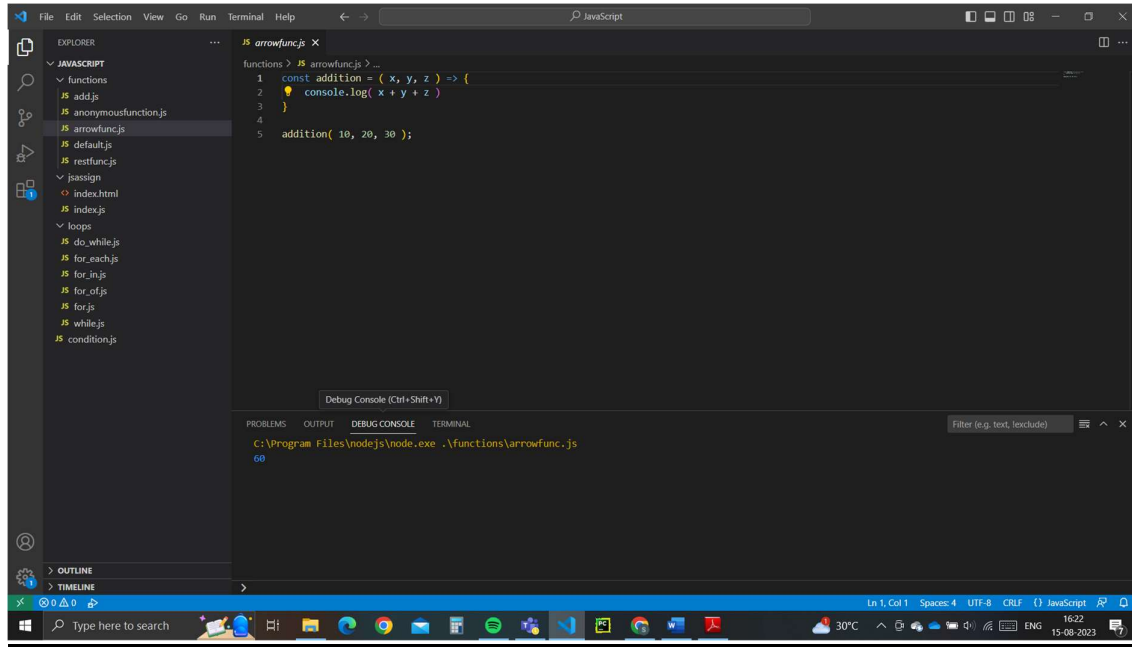


The screenshot shows the Visual Studio Code editor with a JavaScript file named `anonymousfunction.js` open. The code defines an anonymous function and calls it. The terminal output shows the function execution result.

```
1 const x = function(){
2   return "Hello world";
3 }
4 console.log(x());
```

Terminal Output:

```
C:\Program Files\nodejs\node.exe .\functions\anonymousfunction.js
Hello world
```

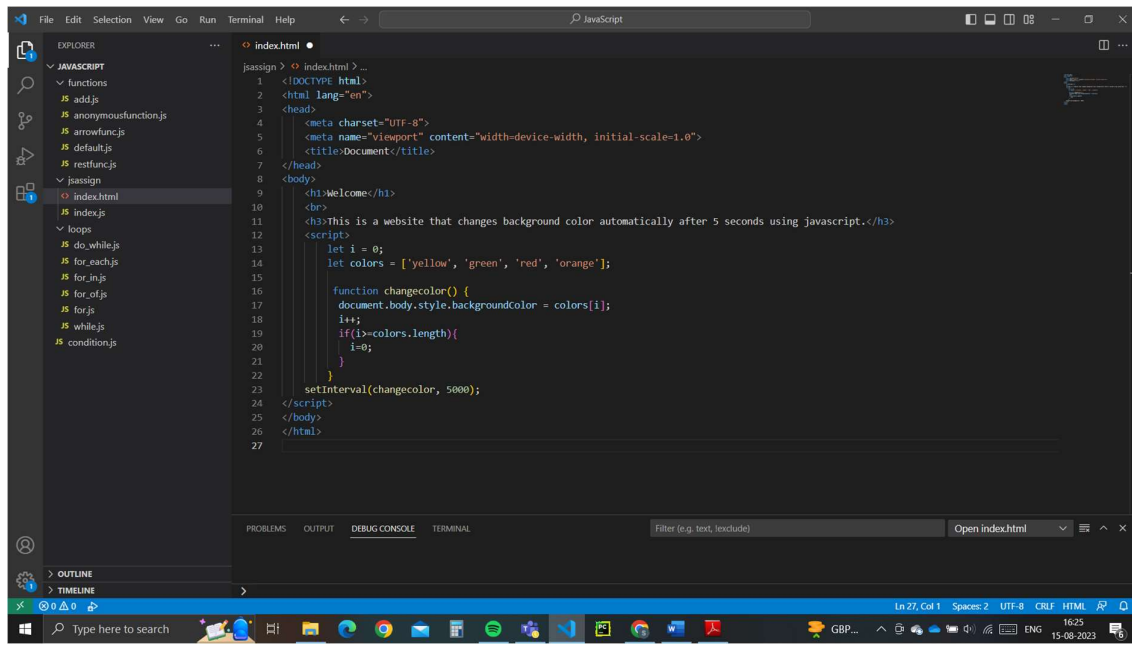
```
functions > # arrowfunc.js > ...
1  const addition = ( x, y, z ) => {
2    console.log( x + y + z )
3  }
4
5  addition( 10, 20, 30 );
```

Debug Console (Ctrl+Shift+Y)

C:\Program Files\nodejs\node.exe .\functions\arrowfunc.js

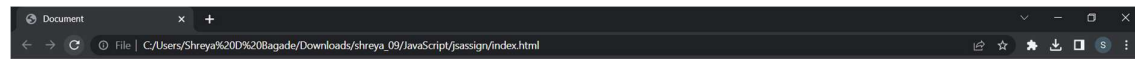
60

Assignment – Change color of web page after 5 seconds.



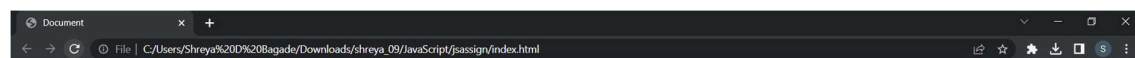
```
jsassign > # index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Document</title>
7  </head>
8  <body>
9    <h1>Welcome</h1>
10   <br>
11   <h3>This is a website that changes background color automatically after 5 seconds using javascript.</h3>
12   <script>
13     let i = 0;
14     let colors = ['yellow', 'green', 'red', 'orange'];
15
16     function changeColor() {
17       document.body.style.backgroundColor = colors[i];
18       i++;
19       if(i==colors.length){
20         i=0;
21       }
22     }
23     setInterval(changeColor, 5000);
24   </script>
25 </body>
26 </html>
27
```

Open index.html



Welcome

This is a website that changes background color automatically after 5 seconds using javascript.



Welcome

This is a website that changes background color automatically after 5 seconds using javascript.



