

# JAVASCRIPT ASSIGNMENT

## Write a function to sum two numbers

### Using + operators

index.js

+

43cuna36j

AI

NEW

JAVASCRIPT

RUN

```
1 let num1 = 10;
2 let num2 = 10;
3 let sum = num1 + num2;
4 console.log("Sum :", sum);
5
```

STDIN

Input for the program ( Optional )

Output:

Sum : 20

### Using function

index.js

+

43cuna36j

AI

NEW

JAVASCRIPT

RUN

```
1 function additionFunction(a, b) {
2     return a + b;
3 }
4
5 let num1 = 5;
6 let num2 = 10;
7 let sum = additionFunction(num1, num2);
8 console.log("Sum of given numbers is:", sum);
9
```

STDIN

Input for the program ( Optional )

Output:

Sum of given numbers is: 15

### Using Arrow function

index.js

+

43cuna36j

AI

NEW

JAVASCRIPT

RUN

```
1 let addition = (a, b) => a + b;
2
3 let num1 = 25;
4 let num2 = 25;
5 let sum = addition(num1, num2);
6 console.log("Sum of given numbers is :", sum);
7
```

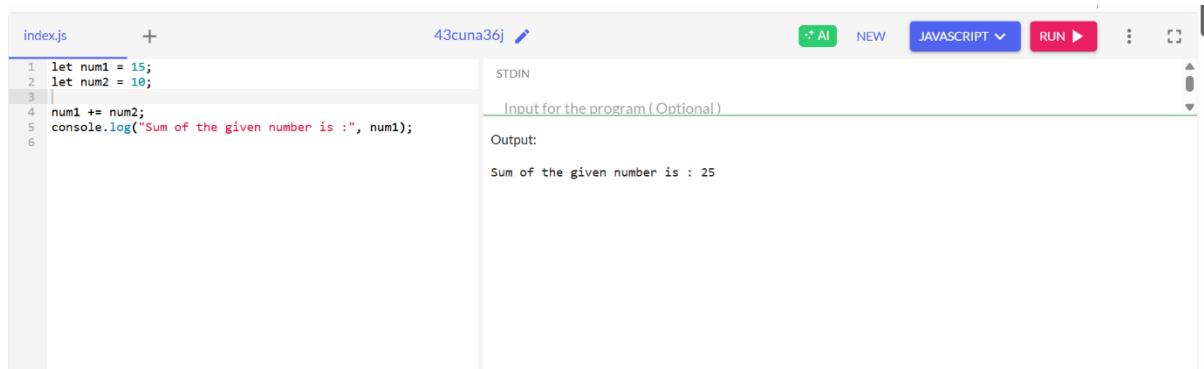
STDIN

Input for the program ( Optional )

Output:

Sum of given numbers is : 50

## Using Addition Assignment operator

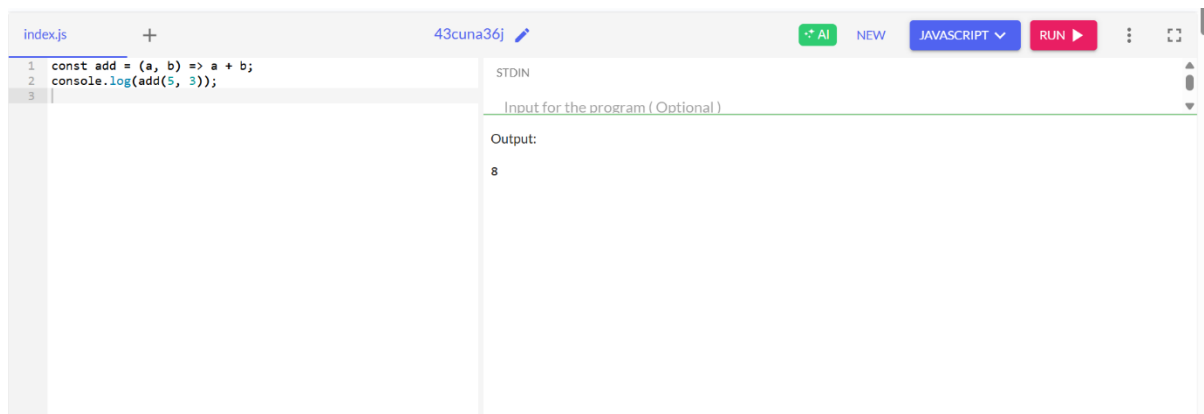


The screenshot shows a code editor with a file named 'index.js'. The code defines two variables, 'num1' and 'num2', with values 15 and 10 respectively. It then uses the addition assignment operator '+= ' to add 'num2' to 'num1'. Finally, it logs the value of 'num1' to the console. The output of the program is 'Sum of the given number is : 25'.

```
1 let num1 = 15;  
2 let num2 = 10;  
3  
4 num1 += num2;  
5 console.log("Sum of the given number is :", num1);  
6
```

Output:  
Sum of the given number is : 25

## Convert a regular function to an arrow function

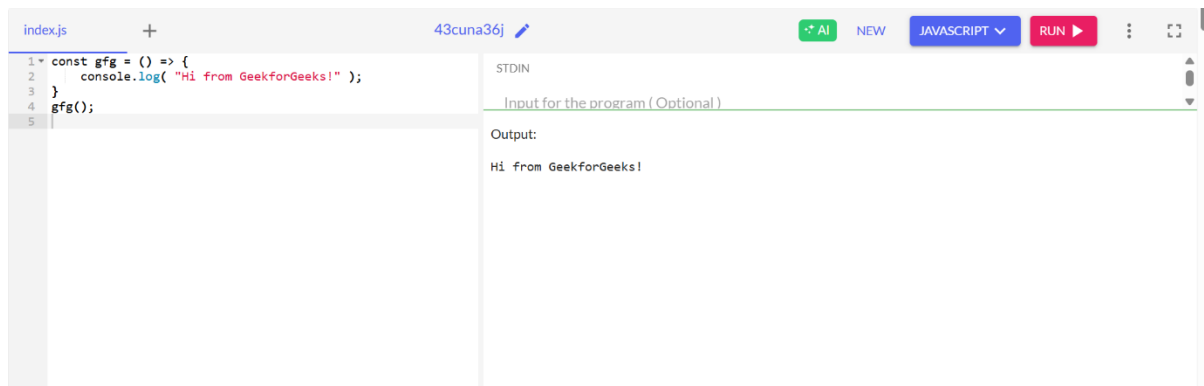


The screenshot shows a code editor with a file named 'index.js'. The code defines a regular function 'add' that takes two arguments, 'a' and 'b', and returns their sum. It then calls the 'add' function with arguments 5 and 3. The output of the program is '8'.

```
1 const add = (a, b) => a + b;  
2 console.log(add(5, 3));  
3
```

Output:  
8

## Arrow function without parameters



The screenshot shows a code editor with a file named 'index.js'. The code defines an arrow function 'gfg' that does not take any arguments and logs the string 'Hi from GeekforGeeks!' to the console. It then calls the 'gfg' function. The output of the program is 'Hi from GeekforGeeks!'.

```
1 const gfg = () => {  
2   console.log("Hi from GeekforGeeks!");  
3 }  
4 gfg();  
5
```

Output:  
Hi from GeekforGeeks!

## Arrow function with single parameters

```
index.js  +  43cuna36j  AI NEW JAVASCRIPT RUN
```

```
1 const square = x => x*x;
2 console.log(square(4));
3
```

STDIN

Input for the program ( Optional )

Output:

16

## Arrow function with multiple parameters

```
index.js  +  43cuna36j  AI NEW JAVASCRIPT RUN
```

```
1 const gfg = ( x, y, z ) => {
2   console.log( x + y + z )
3 }
4 gfg( 10, 20, 30 );
```

STDIN

Input for the program ( Optional )

Output:

60

## Arrow function with default parameters

```
index.js  +  43cuna36j  AI NEW JAVASCRIPT RUN
```

```
1 const gfg = ( x, y, z = 30 ) => {
2   console.log( x + " " + y + " " + z );
3 }
4 gfg( 10, 20 );
5
```

STDIN

Input for the program ( Optional )

Output:

10 20 30

## Return object literals

```
index.js  +  43cuna36j  AI NEW JAVASCRIPT RUN
```

```
1 const makePerson = (firstName, lastName) =>
2   ({first: firstName, last: lastName});
3 console.log(makePerson("Pankaj", "Bind"));
4
```

STDIN

Input for the program (Optional)

Output:

```
{ first: 'Pankaj', last: 'Bind' }
```

## Async arrow function

```
index.js  +  43cuna36j  AI NEW JAVASCRIPT RUN
```

```
1 const fetchData = async () => {
2   const data = await fetch('https://api.example.com/data')
3   return data.json();
4 };
5
```

STDIN

Input for the program (Optional)

Output:

```
URLs found in the code!
You must login to run this program.
```

## Create a counter function using closures

```
index.js  +  43cuna36j  AI NEW JAVASCRIPT RUN
```

```
1 function outer() {
2   let str = "GeeksforGeeks";
3   function inner() {
4     console.log(str);
5   }
6   return inner;
7 }
8 const fun = outer();
9 fun();
10
```

STDIN

Input for the program (Optional)

Output:

```
GeeksforGeeks
```

## Define an object representing a car with properties and a method

index.js

+

43cuna36j

AI

NEW

JAVASCRIPT

RUN

```
1 function vehicle(name, maker, engine) {
2   this.name = name;
3   this.maker = maker;
4   this.engine = engine;
5 }
6
7 let car = new vehicle('GT', 'BMW', '1998cc');
8
9 console.log(car.name);
10 console.log(car.maker);
11 console.log(car['engine']);
12
13
```

STDIN

Input for the program (Optional)

Output:

GT

BMW

1998cc

OneCompiler

PRICING

EDITOR

CHALLENGES

COMPANY & MORE

LOGIN

index.js

+

43cuna36j

AI

NEW

JAVASCRIPT

RUN

```
1 let car = {
2   name: 'GT',
3   maker: 'BMW',
4   engine: '1998cc'
5 };
6
7 console.log(car.name);
8 console.log(car['maker']);
9
```

STDIN

Input for the program (Optional)

Output:

GT

BMW

index.js

+

43cuna36j

AI

NEW

JAVASCRIPT

RUN

```
1 let car = {
2   name: 'GT',
3   maker: 'BMW',
4   engine: '1998cc'
5 };
6
7 car.brakesType = 'All Disc';
8 console.log(car);
9
```

STDIN

Input for the program (Optional)

Output:

{ name: 'GT', maker: 'BMW', engine: '1998cc', brakesType: 'All Disc' }

≡

OneCompiler

PRICINGEDITORCHALLENGESCOMPANY & MORE

LOGIN

index.js43cuna36jNEWJAVASCRIPTRUN

```
1 let car = {
2   name : 'GT',
3   maker : 'BMW',
4   engine : '1998cc',
5   start : function(){
6     console.log('Starting the engine...');
7   }
8 };
9 car.start();
10
11 car.stop = function() {
12   console.log('Applying Brake...');
13 }
14 car.stop();
15
```

STDIN

Input for the program (Optional)

Output:

Starting the engine...  
Applying Brake...

≡

OneCompiler

PRICINGEDITORCHALLENGESCOMPANY & MORE

LOGIN

index.js43cuna36jNEWJAVASCRIPTRUN

```
1 const coder = {
2   isStudying : false,
3   printIntroduction : function(){
4     console.log('My name is ${this.name}. Am I studying?');
5   }
6 };
7 const me = Object.create(coder);
8 me.name = 'Mugil';
9 me.isStudying = true;
10 me.printIntroduction();
11
```

STDIN

Input for the program (Optional)

Output:

My name is Mugil. Am I studying?: true

≡

OneCompiler

PRICINGEDITORCHALLENGESCOMPANY & MORE

LOGIN

index.js43cuna36jNEWJAVASCRIPTRUN

```
1 class Vehicle {
2   constructor(name, maker, engine) {
3     this.name = name;
4     this.maker = maker;
5     this.engine = engine;
6   }
7 }
8
9 let car1 = new Vehicle('GT', 'BMW', '1998cc');
10
11 console.log(car1.name);
12
```

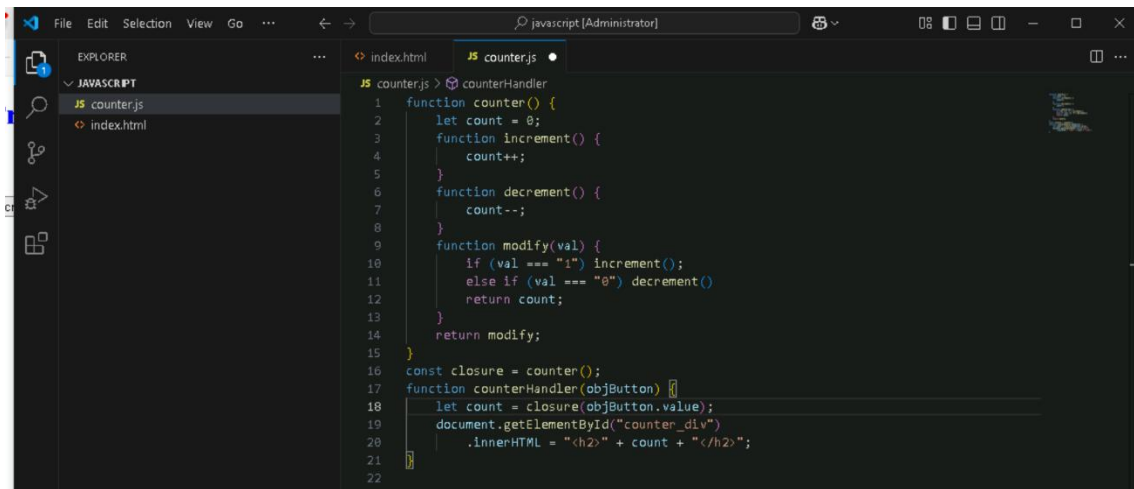
STDIN

Input for the program (Optional)

Output:

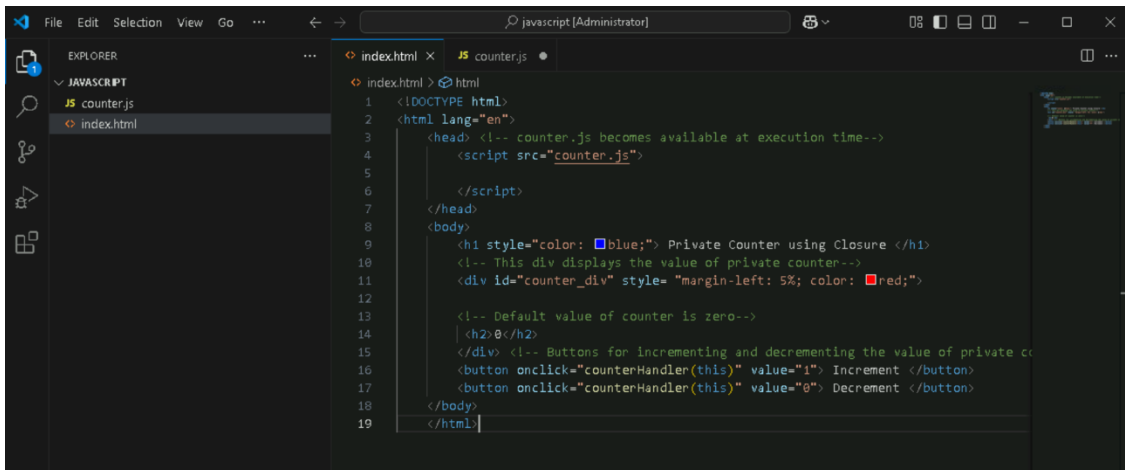
GT

Create a counter function using closures



The image shows a VS Code editor window with the file explorer on the left showing a project named 'JAVASCRIPT' containing 'counter.js' and 'index.html'. The main editor area displays the contents of 'counter.js'. The code defines a 'counter' function that returns a closure 'counterHandler'. The closure has a private 'count' variable and two methods: 'increment' and 'decrement'. A 'modify' function is also defined, which calls either 'increment' or 'decrement' based on the input value. The 'counterHandler' function is then used to create a closure that updates the DOM with the current count.

```
1 function counter() {  
2   let count = 0;  
3   function increment() {  
4     count++;  
5   }  
6   function decrement() {  
7     count--;  
8   }  
9   function modify(val) {  
10    if (val === "1") increment();  
11    else if (val === "0") decrement();  
12    return count;  
13  }  
14  return modify;  
15 }  
16 const closure = counter();  
17 function counterHandler(objButton) {  
18   let count = closure(objButton.value);  
19   document.getElementById("counter_div")  
20     .innerHTML = "<h2>" + count + "</h2>";  
21 }  
22
```

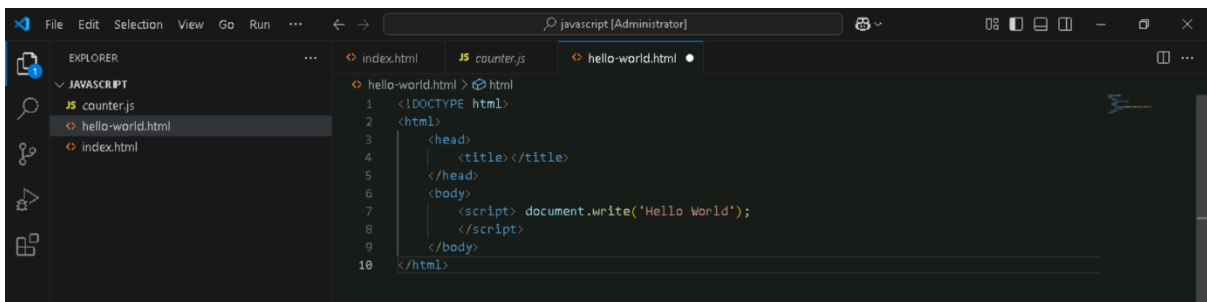


The image shows the same VS Code editor window, but now displaying 'index.html'. The HTML file includes a DOCTYPE declaration, a head section with a script tag for 'counter.js', and a body section. The body contains a heading 'Private Counter using Closure', a comment about the div displaying the value, a div with id 'counter\_div' and style 'margin-left: 5%; color: red;', a comment about the default value, and two buttons: 'Increment' and 'Decrement'. The buttons have onclick events that call 'counterHandler' with '1' and '0' respectively.

```
1 <!DOCTYPE html>  
2 <html lang="en">  
3   <head> <!-- counter.js becomes available at execution time-->  
4     <script src="counter.js">  
5   </script>  
6   </head>  
7   <body>  
8     <h1 style="color: blue;"> Private Counter using Closure </h1>  
9     <!-- This div displays the value of private counter-->  
10    <div id="counter_div" style="margin-left: 5%; color: red;">  
11      <!-- Default value of counter is zero-->  
12      <h2>0</h2>  
13    </div> <!-- Buttons for incrementing and decrementing the value of private counter-->  
14    <button onclick="counterHandler(this)" value="1"> Increment </button>  
15    <button onclick="counterHandler(this)" value="0"> Decrement </button>  
16  </body>  
17 </html>
```

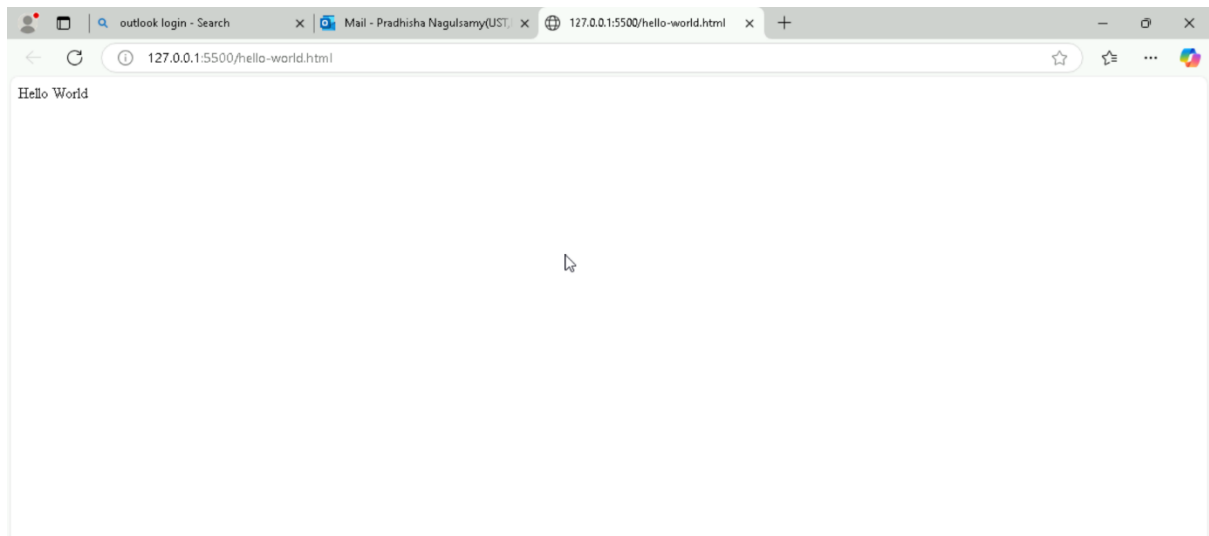


Create a simple web page displaying "Hello, World!" using JavaScript.



The image shows the VS Code editor window with the file explorer on the left showing a project named 'JAVASCRIPT' containing 'counter.js', 'hello-world.html', and 'index.html'. The main editor area displays the contents of 'hello-world.html'. The code is a simple HTML document with a title 'Hello World' and a script tag that calls 'document.write' to display 'Hello World!' in the browser.

```
1 <!DOCTYPE html>  
2 <html>  
3   <head>  
4     <title></title>  
5   </head>  
6   <body>  
7     <script> document.write('Hello World!');  
8   </script>  
9   </body>  
10 </html>
```



## Pop-Up on Client Side

