**Umer Khan Assessment**

**1.Write a function that prints the numbers from 1 to 100. But for multiples of three, print "Fizz" instead of the number, and for the multiples of five, print "Buzz". For numbers that are multiples of both three and five, print "FizzBuzz".  
Solution:**

**function printFizzBuzzNumbers() {**

***for* (let i = 1; i <= 100; i++) {**

***if* (i % 3 == 0) {**

**console.log("Fizz");**

**}**

***if* (i % 5 == 0) {**

**console.log("Bizz");**

**}**

***if* (i % 3 == 0 && i % 5 == 0) {**

**console.log("FizzBuzz");**

**} *else* {**

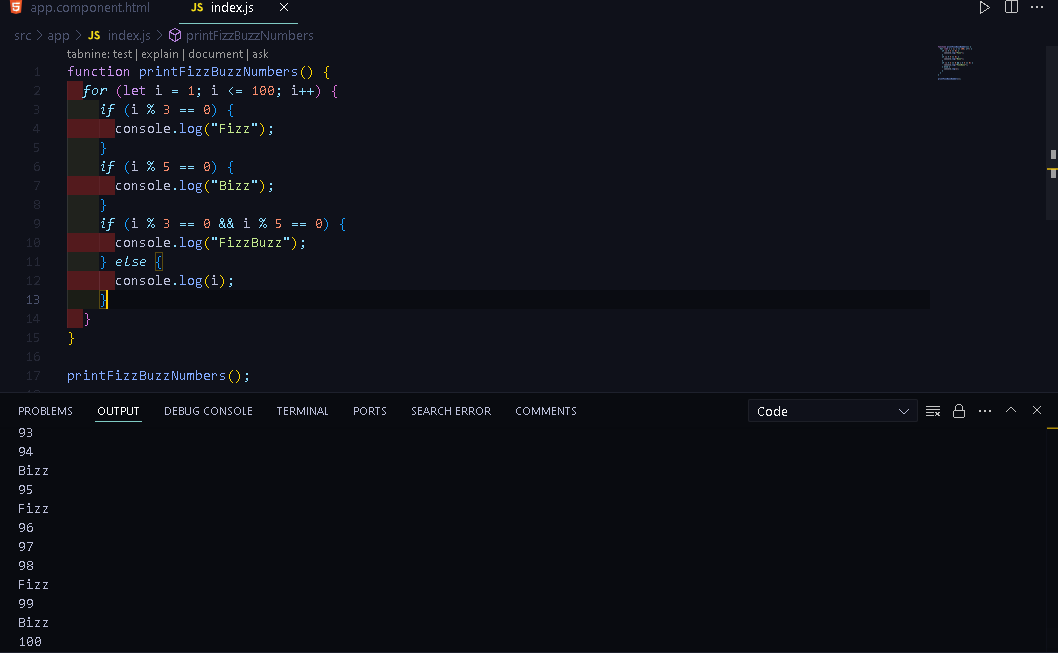
**console.log(i);**

**}**

**}**

**}**

**printFizzBuzzNumbers();**

****

**Q1 Snap Shot**

**2.Write a function that takes a string input representing a simple arithmetic expression (only addition and subtraction) and returns the result.**

**Solution:**

**function StringOperation(*StringArgument*) {**

***StringArgument* = *StringArgument*.replace(/\s+/g, "");**

**let total = 0;**

**let currentNumber = 0;**

**let currentOperator = "+";**

***for* (let i = 0; i < *StringArgument*.length; i++) {**

**let Argument = *StringArgument*[i];**

***if* (!isNaN(Argument)) {**

**currentNumber = currentNumber \* 10 + parseInt(Argument);**

**} *else* *if* (Argument === "+" || Argument === "-") {**

***if* (currentOperator === "+") {**

**total += currentNumber;**

**} *else* *if* (currentOperator === "-") {**

**total -= currentNumber;**

**}**

**currentOperator = Argument;**

**currentNumber = 0;**

**}**

**}**

***if* (currentOperator === "+") {**

**total += currentNumber;**

**} *else* *if* (currentOperator === "-") {**

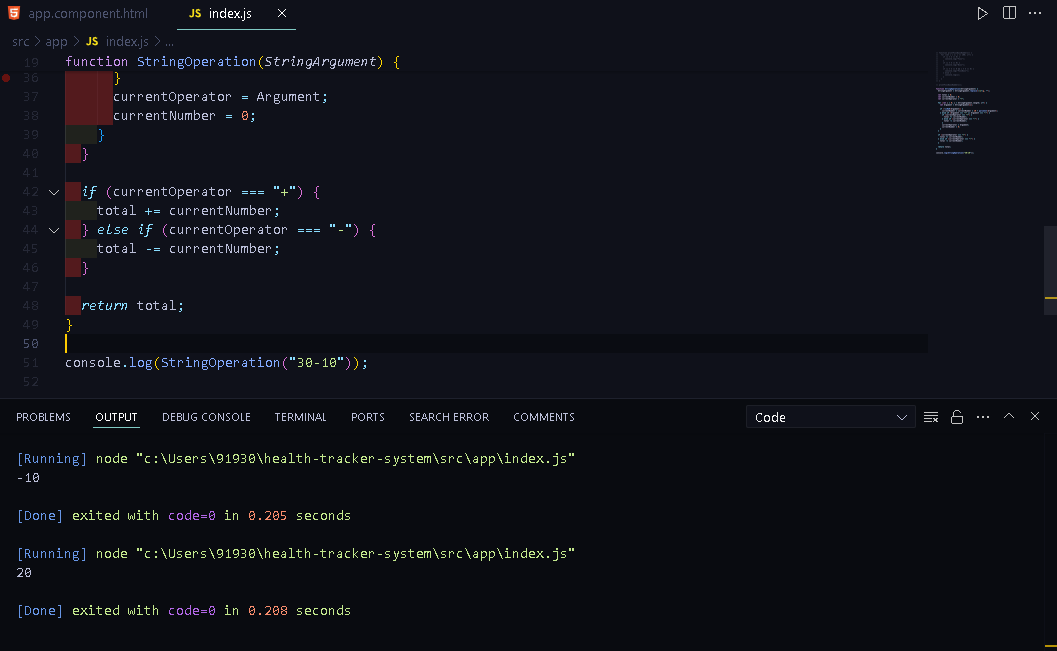
**total -= currentNumber;**

**}**

***return* total;**

**}**

**console.log(StringOperation("30-10"));**

****

**Q2 Snap Shot**

**3.Write a function that takes a nested array and returns a flattened array.  
Solution:**

**function ArrayFlatneer(*nestedArray*) {**

**let result = [];**

***nestedArray*.forEach((*item*) => {**

***if* (Array.isArray(*item*)) {**

**result = result.concat(ArrayFlatneer(*item*));**

**} *else* {**

**result.push(*item*);**

**}**

**});**

***return* result;**

**}**

**function ArrayFlatneerwithFlatfun(*nestedArray*) {**

***return* *nestedArray*.flat(Infinity);**

**}**

**const nestedname = ["U", ["M", ["A", "R"], "K"], "H", ["A", "N"]];**

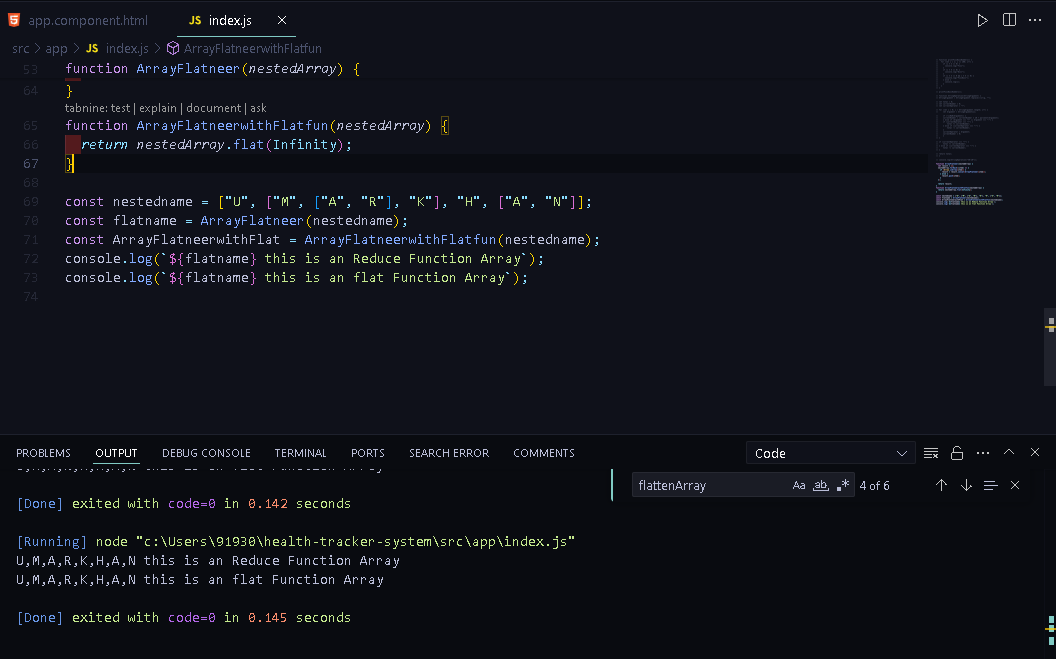
**const flatname = ArrayFlatneer(nestedname);**

**const ArrayFlatneerwithFlat = ArrayFlatneerwithFlatfun(nestedname);**

**console.log(`${flatname} this is an Reduce Function Array`);**

**console.log(`${flatname} this is an flat Function Array`);**

**Note: I have Implemented two functions to Achieve the Functionally.**

****

**Q3 Snap Shot**

**4.Write a function that checks if two given strings are anagrams of each other.**

**Solution:**

**function AreEqualinletter(*str1*, *str2*) {**

**const cleanStrOne = *str1*.replace(/[^\w]/g, "").toLowerCase();**

**const cleanStrTwo = *str2*.replace(/[^\w]/g, "").toLowerCase();**

***if* (cleanStrOne.length !== cleanStrTwo.length) {**

***return* false + " Your Both String Length Dosen't Match with Eachother";**

**} *else* {**

**const sortedStrOne = cleanStrOne.split("").sort().join("");**

**const sortedStrTwo = cleanStrTwo.split("").sort().join("");**

***if* (sortedStrOne === sortedStrTwo) {**

***return* true + " Your Both String is anagrams of each other";**

**} *else* {**

***return* false + " Your Both String is not anagrams of each other";**

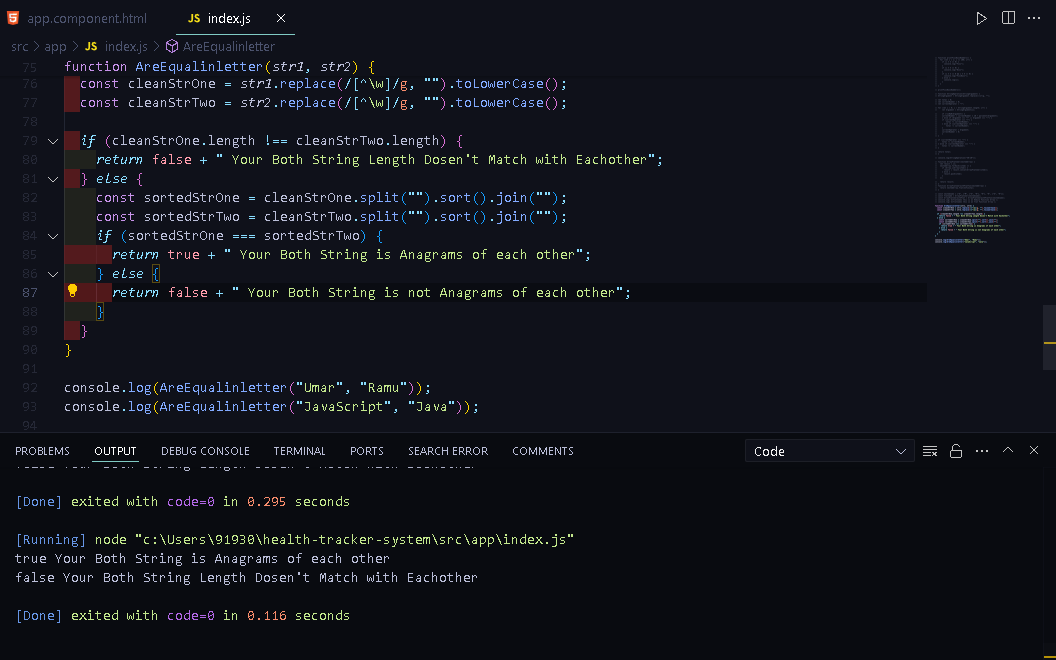
**}**

**}**

**}**

**console.log(AreEqualinletter("Umar", "Ramu"));**

**console.log(AreEqualinletter("JavaScript", "Java"));**



**Q4 Snap Shot**

**5.Write a function that takes an array and returns a new array with duplicates removed.**

**Solution:**

**function RemovedublicateElement(*OgArray*) {**

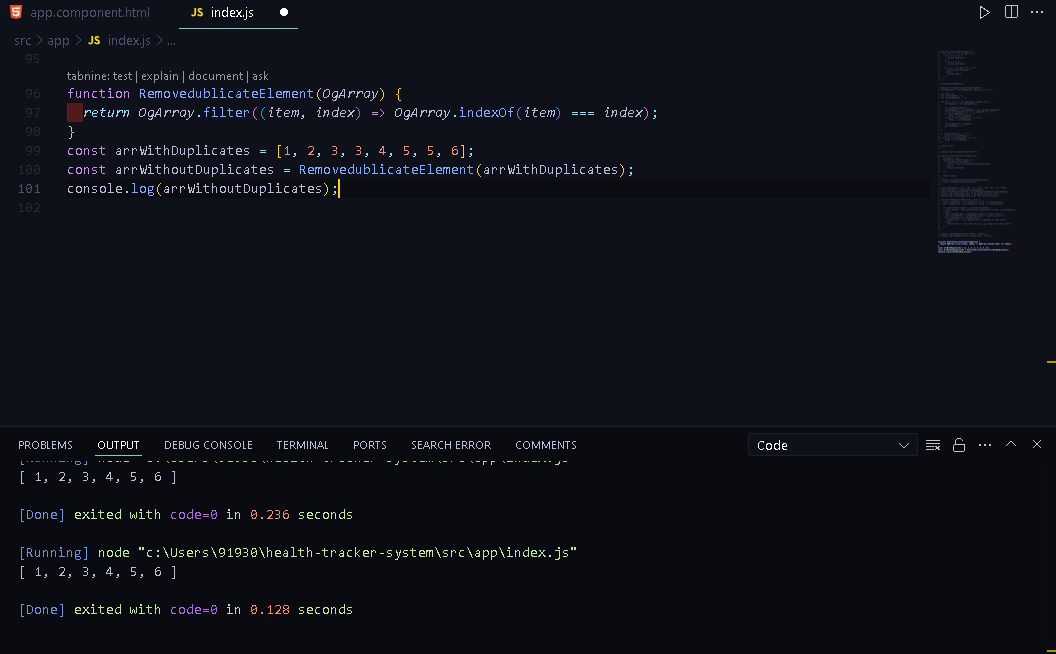
***return* *OgArray*.filter((*item*, *index*) => *OgArray*.indexOf(*item*) === *index*);**

**}**

**const arrWithDuplicates = [1, 2, 3, 3, 4, 5, 5, 6];**

**const arrWithoutDuplicates = RemovedublicateElement(arrWithDuplicates);**

**console.log(arrWithoutDuplicates);**

****

**Q5 Snap Shot**

**6.Write a function that takes a string and capitalizes the first letter of each word in the string.**

**Solution:**

**function FirstwordCapitilizer(*string*) {**

**const words = *string*.split(" ");**

**const capitalizedWords = words.map(**

**(*word*) => *word*.charAt(0).toUpperCase() + *word*.slice(1)**

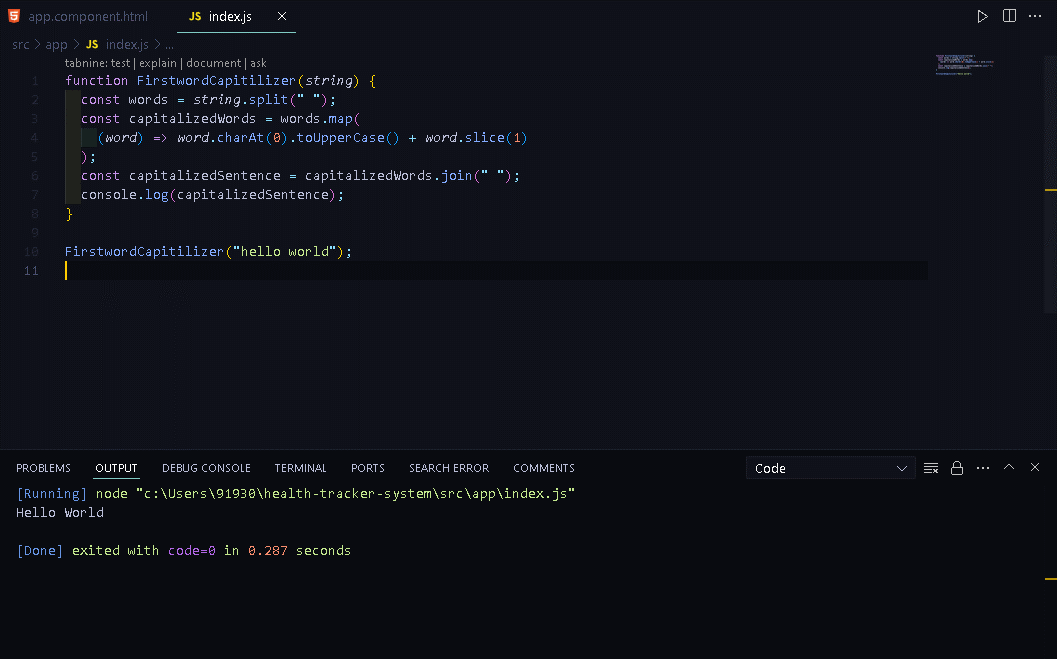
**);**

**const capitalizedSentence = capitalizedWords.join(" ");**

**console.log(capitalizedSentence);**

**}**

**FirstwordCapitilizer("hello world");**



**Q6 Snap Shot**

**7.Write a function that generates the first n numbers of the Fibonacci sequence**

**Solution:**

**function generateNFibonacci(*n*) {**

***if* (*n* <= 0) *return* [];**

***if* (*n* === 1) *return* [0];**

**let fibonacciNumbers = [0, 1];**

***for* (let i = 2; i < *n*; i++) {**

**fibonacciNumbers[i] = fibonacciNumbers[i - 1] + fibonacciNumbers[i - 2];**

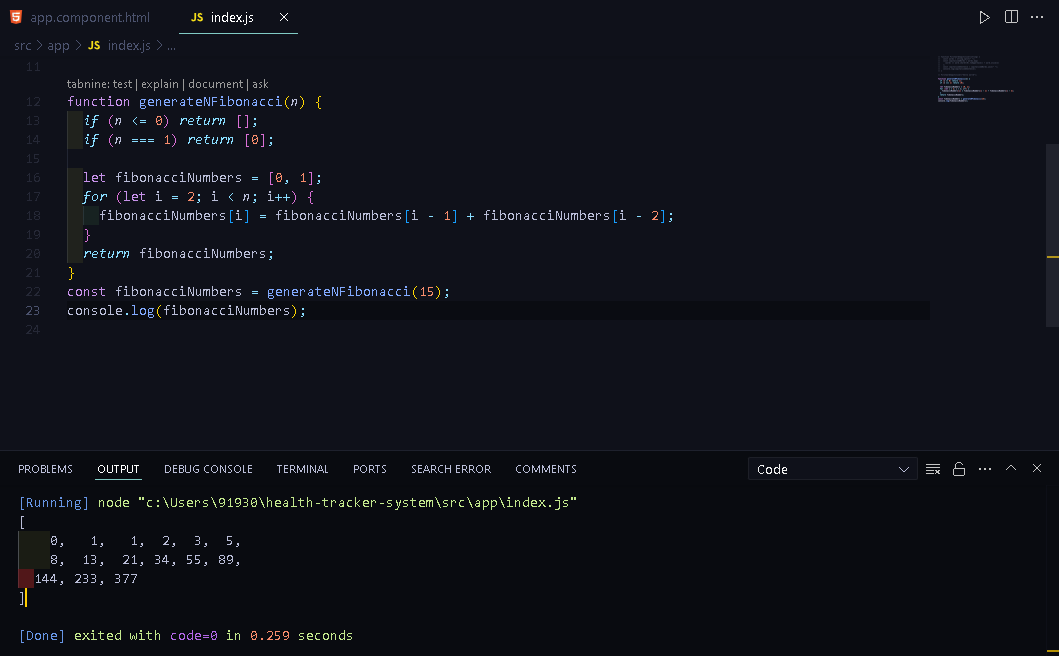
**}**

***return* fibonacciNumbers;**

**}**

**const fibonacciNumbers = generateNFibonacci(15);**

**console.log(fibonacciNumbers);**

****

**Q7 Snap Shot**

**8.Implement a simple HashMap class with put, get, and remove methods.**

**Solution:**

**class HashMapClass {**

**constructor() {**

**this.map = new Map();**

**}**

**put(*key*, *value*) {**

**this.map.set(*key*, *value*);**

**}**

**get(*key*) {**

***return* this.map.get(*key*);**

**}**

**remove(*key*) {**

**this.map.delete(*key*);**

**}**

**}**

**const myHashMap = new HashMapClass();**

**myHashMap.put("name", "Umer");**

**myHashMap.put("age", 21);**

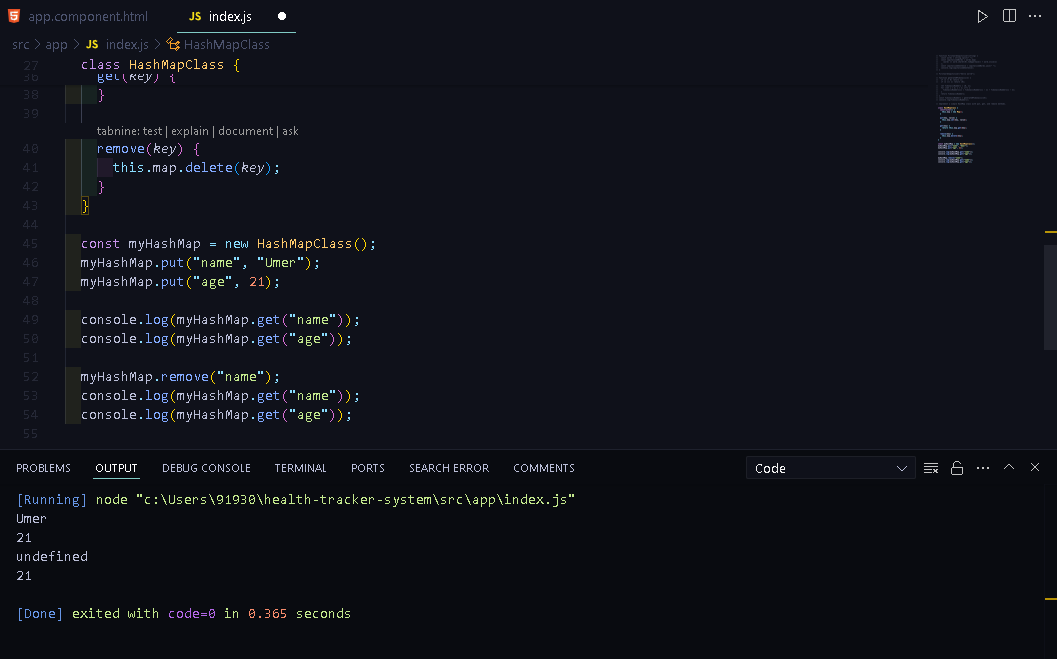
**console.log(myHashMap.get("name"));**

**console.log(myHashMap.get("age"));**

**myHashMap.remove("name");**

**console.log(myHashMap.get("name"));**

**console.log(myHashMap.get("age"));**

****

**Q8 Snap Shot**

**9.Write a function that filters out even numbers from an array.**

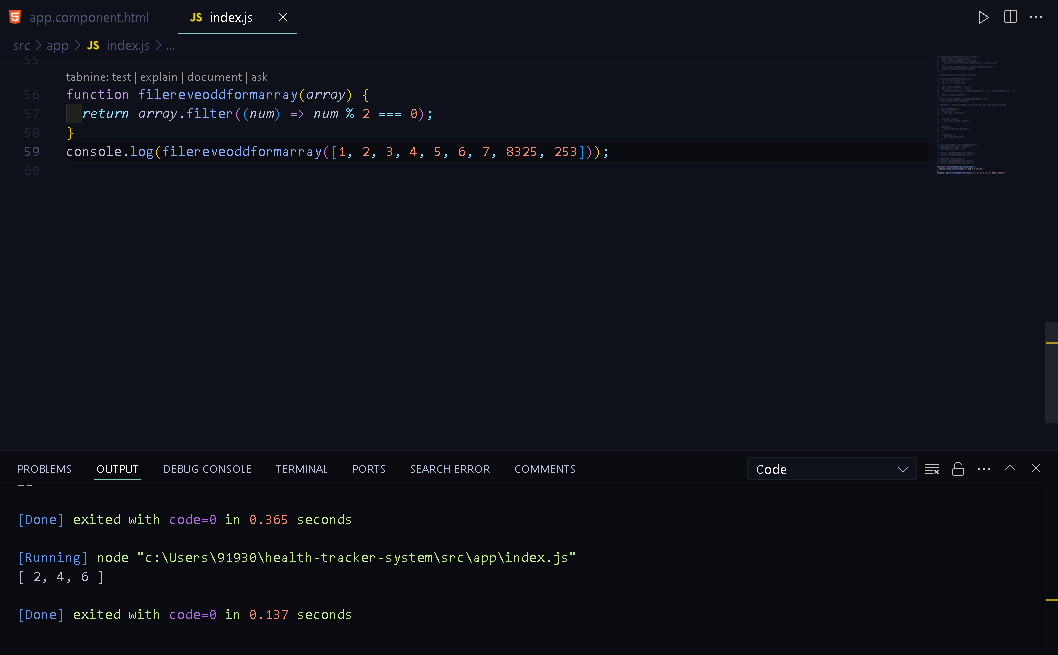
**Solution:**

**function filereveoddformarray(*array*) {**

***return* *array*.filter((*num*) => *num* % 2 === 0);**

**}**

**console.log(filereveoddformarray([1, 2, 3, 4, 5, 6, 7, 8325, 253]));**

****

**Q9 Snap Shot**

**10.Write a function that converts a given string to title case (capitalizing the first letter of each word).**

**Solution:**

**function titleCase(*str*) {**

**const words = *str*.toLowerCase().split(" ");**

**const capitalizedWords = words.map(**

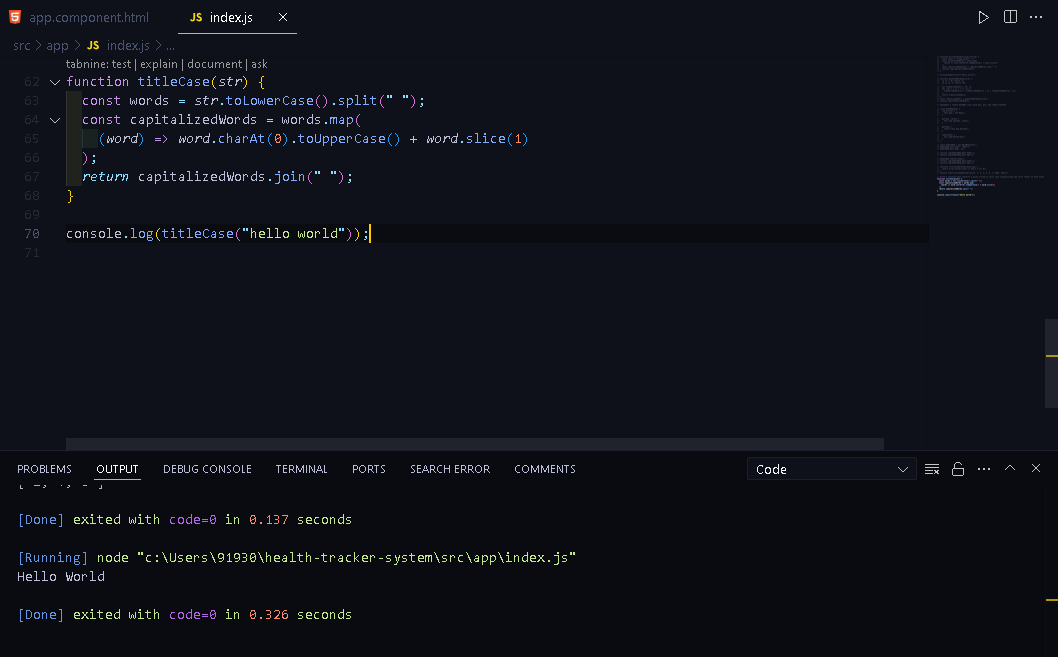
**(*word*) => *word*.charAt(0).toUpperCase() + *word*.slice(1)**

**);**

***return* capitalizedWords.join(" ");**

**}**

**console.log(titleCase("hello world"));**



**Q10 Snap Shot**