

# ASSIGNMENT

## Microservices

Name: Aditya Shrivastava

Class: Section – E

Roll Number: 10721210002

Q.1 Write a function that takes an array of words and maps each word to its length. Provide a sample array and the resulting array of word lengths.

Code :

```
index.js  [icon] × +
index.js > f mapWordsToLength > ...
1  function mapWordsToLength (words) {
2
3  return words.map(word => word.length);
4  }
5  const sampleArray = ["apple", "banana", "orange", "grape"];
6  const resultArray = mapWordsToLength(sampleArray);
7  console.log(resultArray);
8
```

Output:

```
>_ Console [icon] × [icon] Shell [icon] × +
Run
[ 5, 6, 6, 5 ]
```

Q.2 Create a function that converts an array of temperatures in Celsius to Fahrenheit using the map function. Display the original array and the array with converted temperatures.

Code:



```
index.js 25 × +
index.js > f celsiusToFahrenheit > ...
1 function celsiusToFahrenheit (celsiusTemps) {
2   return celsiusTemps.map(celsius => (celsius * 9/5) + 32);
3 }
4 const celsiusTemperatures = [0, 15, 25, 30, 10];
5 const fahrenheitTemperatures = celsiusToFahrenheit (celsiusTemperatures);
6 console.log("Original Celsius Temperatures:", celsiusTemperatures);
7 console.log("Fahrenheit Temperatures:", fahrenheitTemperatures);
```

Output:



```
>_ Console × Shell × +
Run
Original Celsius Temperatures: [ 0, 15, 25, 30, 10 ]
Fahrenheit Temperatures: [ 32, 59, 77, 86, 50 ]
```

Q.3 Design a function that maps numerical grades to letter grades (e.g., 90 to 'A', 80 to 'B', and so on). Provide an array of numerical grades and the resulting array of mapped letter grades.

Code:

```
index.js × index.js × +
index.js > f mapNumericalToLetterGrades > ...
1 function mapNumericalToLetterGrades (numericalGrades) {
2 return numericalGrades.map(grade => {
3 if (grade >= 90) { return 'A';
4 } else if (grade >= 80) { return 'B';
5 } else if (grade >= 70) { return 'C';
6 } else if (grade >= 60) {
7 return 'D';
8 } else {
9 return 'F';
10 }
11 });
12 }
13
14 const numericalGradesArray = [95, 87, 72, 60, 88];
15 const letterGradesArray = mapNumericalToLetterGrades (numericalGradesArray);
16 console.log("Numerical Grades:", numericalGradesArray);
17 console.log("Letter Grades:", letterGradesArray);
18
19
```

Output:

```
>_ Console × Shell × +
Run
Numerical Grades: [ 95, 87, 72, 60, 88 ]
Letter Grades: [ 'A', 'B', 'C', 'D', 'B' ]
```

Q.4 Implement a function that filters out prime numbers from an array of integers using the filter function. Provide a sample array and the resulting array with only prime numbers.

Code:

```
index.js x index.js x +
index.js > f isPrime > ...
1 ✓ function isPrime(num) {
2   if (num <= 1) return false;
3 ✓ for (let i = 2; i <= Math.sqrt(num); i++){
4   if (num % i === 0) return false;
5 }
6   return true;
7 }
8
9 ✓ function filterOutPrimes (numbers) {
10 return numbers.filter(number => isPrime(number));
11 }
12 const sampleArray = [2, 5, 8, 11, 15, 20, 23, 29];
13 const primeNumbersArray = filterOutPrimes(sampleArray);
14 console.log("Original Array:", sampleArray);
15 18
16 19
17 20
18 console.log("Prime Numbers:", primeNumbersArray);
```

Output:

```
>_ Console x Shell x +
  Run
Original Array: [
  2, 5, 8, 11,
  15, 20, 23, 29
]
Prime Numbers: [ 2, 5, 11, 23, 29 ]
```

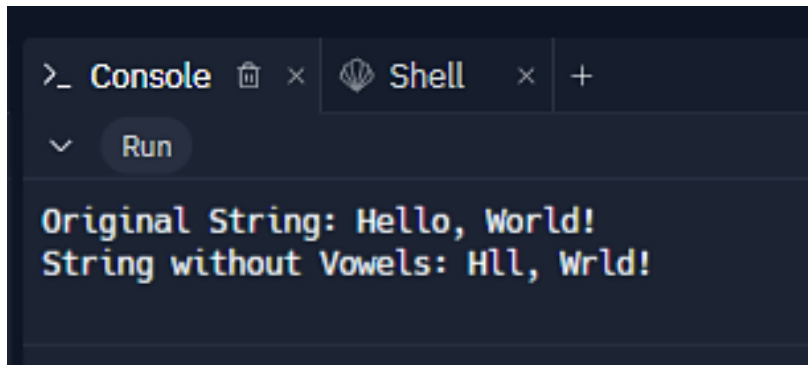
Q.5 Write a function that filters out vowels from a given string using the filter function. Show a sample string and the resulting string without vowels.

Code:



```
index.js x index.js x +
index.js > ...
1 function filterOutVowels (inputString) {
2
3   const vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'];
4   return inputString.split('').filter(char => !vowels.includes(char)).join('');
5 }
6 const sampleString = "Hello, World!";
7 const stringWithoutVowels = filterOutVowels (sampleString);
8 console.log("Original String:", sampleString);
9 console.log("String without Vowels:", stringWithoutVowels);
```

Output:



```
>_ Console x Shell x +
Run
Original String: Hello, World!
String without Vowels: Hll, Wrld!
```

Q.6 Create a function that filters out numbers divisible by 5 from an array of integers using the filter function. Provide a sample array and the resulting filtered array.

Code:

```
index.js × index.js × +
index.js > ...
1 function filterOutDivisibleBy5 (numbers) {
2
3   return numbers.filter(number => number % 5 !== 0);
4 }
5 const sampleArray = [2, 5, 10, 15, 7, 20, 25, 30];
6 const filteredArray = filterOutDivisibleBy5(sampleArray);
7 console.log("Original Array:", sampleArray);
8
9 console.log("Filtered Array:", filteredArray);
```

Output:

```
>_ Console × Shell × +
Run
Original Array: [
  2, 5, 10, 15,
  7, 20, 25, 30
]
Filtered Array: [ 2, 7 ]
```

Q.7 Create a function that calculates the factorial of a given number using the reduce function. Display the calculated factorial of a specific number.

Code:

```
index.js x index.js x +
index.js > ...
1 function calculateFactorial (number) {
2   if (number < 0) {
3     return "Factorial not defined for negative numbers.";
4   } else if (number == 0 || number == 1) {
5     return 1;
6   } else {
7     return Array.from({ length: number }, (_, index) => index + 1).reduce((acc, curr) => acc*curr);
8   }
9 }
10
11 const numberToCalculateFactorial = 6;
12
13 const factorialResult = calculateFactorial(numberToCalculateFactorial);
14
15 console.log(`Factorial of ${numberToCalculateFactorial} is: ${factorialResult}`);
16
```

Output:

```
>_ Console x Shell x +
Run
Factorial of 6 is: 720
```



Q.8 Implement a function to find the maximum number in an array using the reduce function. Provide a sample array and display the maximum number from the array.

Code:

```
index.js x index.js x +
index.js > ...
1 function findMaxNumber (numbers) {
2   return numbers.reduce((max, current) => (current > max ? current : max), -Infinity);
3 }
4
5 const sampleArray = [3, 8, 12, 5, 20, 7, 15];
6 const maxNumber = findMaxNumber(sampleArray);
7 console.log("Original Array:", sampleArray);
8 console.log("Maximum Number:", maxNumber);
9
```

Output:

```
>_ Console x Shell x +
Run
Original Array: [
  3, 8, 12, 5,
  20, 7, 15
]
Maximum Number: 20
```

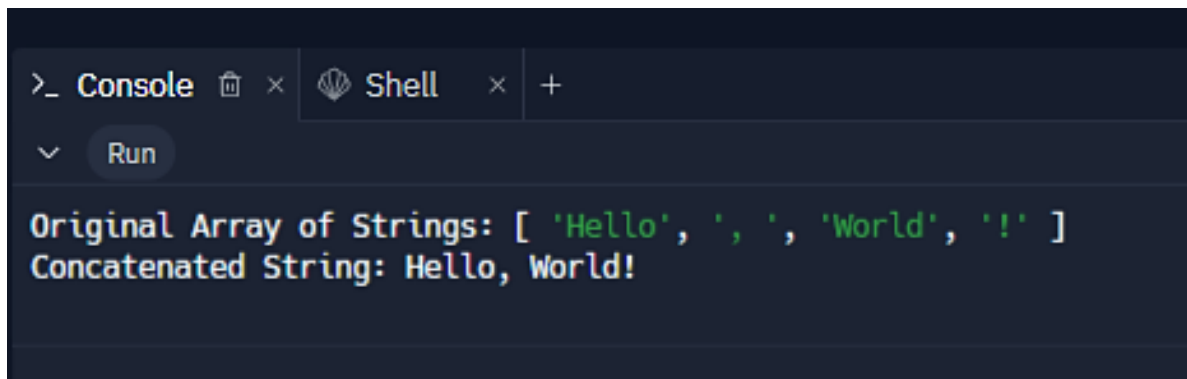
Q.9 Write a function that concatenates an array of strings into a single string using the reduce function. Display the original array of strings and the resulting concatenated string.

Code:



```
1 function concatenateStrings (strings) {  
2   return strings.reduce((concatenatedString, currentString) => concatenatedString + currentString, '');  
3 }  
4 const stringArray = ["Hello", ", ", "World", "!"];  
5 const concatenatedString = concatenateStrings(stringArray);  
6 console.log("Original Array of Strings:", stringArray);  
7 console.log("Concatenated String:", concatenatedString);  
8
```

Output:



```
> Console Shell +  
Run  
Original Array of Strings: [ 'Hello', ', ', 'World', '!' ]  
Concatenated String: Hello, World!
```