Practices

Practice

- Write a **function addProperty(obj, key, value)** that adds a new property to an object and returns the updated object.

- **Output**
  - console.log(addProperty({ name: 'John' }, 'age', 25))
    // { name: 'John', age: 25 }
  - console.log(addProperty({}, 'city', 'New York’))
    // { city: 'New York' }

Practice

- Write a **function mergeObject(obj1, obj2)** and return merges two objects.

- **Output**
  - console.log(mergeObject({ name: 'John', age: 25 }, { city: 'New York' }))
    //{ name: 'John', age: 25, city: 'New York' }
  - console.log(mergeObject({ name: 'Jane' }, { age: 30, country: 'USA' }))
    //{ name: 'Jane', age: 30, country: 'USA' }

Practice

- Write **a function getFreqOfWords(sentence)** that returns an object with keys representing unique words in lowercase and values representing the frequency of occurrences of each word with ignore case in the sentence. If the input string is null or undefined, return undefined.

- **Output**
  - console.log(getFreqOfWords('Today is present and present is your gift'))

  //{ today: 1, is: 2, present: 2, and: 1, your: 1, gift: 1 }
  - console.log(getFreqOfWords('Do you best just do it’))

  //{ do: 2, you: 1, best: 1, just: 1, it: 1 }
  - console.log(getFreqOfWords(null)) //undefined
  - console.log(getFreqOfWords(undefined)) //undefined

Practice

- Write a **function extractAndRename(obj)** that extracts properties name and age from an object, renames them to fullName and yearsOld, and returns a new object with these properties.

- **Output**
  - console.log(extractAndRename({ name: 'John', age: 25, city: 'New York' }))
  - // { fullName: 'John', yearsOld: 25 }
  - console.log(extractAndRename({ name: 'Jane', age: 30 }))
  - // { fullName: 'Jane', yearsOld: 30 }

Practice

- Write a **function mergeAndDestructure(obj1, obj2)** that merges two objects using the spread operator, and then destructures the merged object to extract specific properties. Return an object with the extracted properties.

- **Output**
  - console.log(mergeAndDestructure({ name: 'John', age: 25 }, { city: 'New York' }))
    // { name: 'John', age: 25 }
  - console.log(mergeAndDestructure({ name: 'Jane' }, { age: 30, country: 'USA' }))
    // { name: 'Jane', age: 30 }

Practice

- Write **a function removeDuplicateWords(sentence)** that takes a string as input and returns a new string with all duplicate words removed, while preserving the order of the first occurrence of each word. The function should be case insensitive, but the output should maintain the original casing. If the input is null, undefined, or an empty string, return an empty string.

- Output:

```
removeDuplicateWords("This is a test This test is easy.")  // "This is a test easy."
removeDuplicateWords("Hello hello HELLO world!") // "Hello world!"
removeDuplicateWords("One two two three three three") // "One two three"
removeDuplicateWords("") // ""
removeDuplicateWords(null) // ""
```

Practice

- Write a function **findValue(arr)** that takes an array of numbers and returns an object with the following properties:
  - min: the minimum value in the array
  - max: the maximum value in the array
  - sum: the sum of all the values in the array
  - avg: the average of all the values in the array

For example, given the array [1, 2, 3, 4, 5], the function should return:

{ min: 1, max: 5, sum: 15, avg: 3 }

Practice

- Write a function **convertToUppercase(arr)** that takes an array of strings and returns a new array with all strings converted to uppercase.

- **Sample Outputs**
  - convertToUppercase(['hello', 'world'])) // ['HELLO', 'WORLD']
  - convertToUppercase(['javascript', 'is', 'fun'])) // ['JAVASCRIPT', 'IS', 'FUN']

Practice

- Write a function **findMax(arr)** that takes an array of numbers and returns the maximum value using the reduce method.

Sample Outputs

- console.log(findMax([1, 2, 3, 4])) // 4
- console.log(findMax([10, 5, 8, 7])) // 10

Practice

- Write a function **removeFalsyValues(arr)** that takes an array and returns a new array with all falsy values removed. Falsy values include 0, empty string, false, null, and undefined.

Sample Outputs

- removeFalsyValues([0, 1, false, 2, '', 3]) // [1, 2, 3]
- removeFalsyValues(['a', '', 'b', null, 'c']) // ['a', 'b', 'c’]
- removeFalsyValues([null, undefined, 'A']) // ['A']

Practice

- Write a function **doubleAndFilterEvenNumbers(arr)** that takes an array of numbers, filter out odd numbers and returns a new array with doubled numbers.

- **Sample Outputs**
  - doubleAndFilterEvenNumbers([1, 2, 3, 4])) // [4, 8]
  - doubleAndFilterEvenNumbers([5, 6, 7, 8])) // [12, 16]

Practice

- Write a function **filterEvenNumbers(arr)** that takes an array of numbers and returns a new array with only the even numbers.

- **Sample Outputs**
  - filterEvenNumbers([1, 2, 3, 4]) // [2, 4]
  - filterEvenNumbers([5, 6, 7, 8]) // [6, 8]

Practice

- Write a function **populateArray(length, value)** that creates an array of given length and fills it with the specified value.

- **Sample Outputs**
  - populateArray(3, 'a') // ['a', 'a', 'a']
  - populateArray(5, 0) // [0, 0, 0, 0, 0]

Practice

- Write a function **squareNumbers(arr)** that takes an array of numbers and returns a new array with each number squared.

- **Sample Outputs**
  - squareNumbers([1, 2, 3, 4]) // [1, 4, 9, 16]
  - squareNumbers([5, 6]) // [25, 36]

Practice

- Write a function **processNumbers(arr)** that takes an array of numbers, squares each number, filters out the even numbers, and returns the sum of the remaining numbers.

- **Sample Outputs**
  - processNumbers([1, 2, 3, 4]) // 10 (1^2 + 3^2)
  - processNumbers([5, 6, 7, 8]) // 74 (5^2 + 7^2)