Functions Practice Master Document — Step-By-Step Logic + Full Explanations

This document is designed so that students can: - Read the **problem** - Try solving it on their own - Compare their attempt with a **step-by-step explanation** - Understand *why every line, operator, and loop exists* - Finally see a **complete function solution**

This makes the learning independent, practical, and beginner-friendly.

---

1. Prime Number Checker

Input

A single number. Example: `17`

Output

`true` (if prime) or `false` (if not prime)

Problem Statement

Write a function that checks whether a number is a prime number.

Step-by-Step Explanation

1. A prime number is a number **greater than 1**.
2. A prime number should be divisible only by **1 and itself**.
3. If a number has any other divisor, it is **not prime**.
4. To check this, we test numbers from **2 up to num/2**.
5. If any of these numbers divide `num` completely (`num % i === 0`), we know it's NOT prime.
6. If no number divides it, then it *is* prime.

Why not check all numbers from 1 to num?

- Unnecessary.
- Any divisor will appear before `num/2`.

Final Function

```
function isPrime(num) {
  // 1. Prime numbers must be greater than 1
```

```
  if (num < 2) return false;

  // 2. Check if any number divides 'num'
  for (let i = 2; i <= num / 2; i++) {
    // 3. If divisible, it's not prime
    if (num % i === 0) return false;
  }

  // 4. If no divisors found, it's prime
  return true;
}
```

---

2. Reverse a String

Input

A string. Example: `"hello"`

Output

Reversed string. Example: `"olleh"`

Problem Statement

Reverse the characters in a string.

Step-by-Step Explanation

1. Strings are collections of characters.
2. We want the last character to become the first.
3. Start from the **end of the string**.
4. Add each character one by one into a new string.
5. Continue until reaching index `0`.

Why loop from last to first?

- Because reversing means flipping the order.

Final Function

```
function reverseString(str) {
  // Create an empty string to store reversed result
  let result = "";
```

```
  // Loop from last index to first
  for (let i = str.length - 1; i >= 0; i--) {
    result += str[i];
  }

  return result;
}
```

---

3. Count Vowels

Input

A string. Example: `"education"`

Output

Number of vowels. Example: `5`

Problem Statement

Count number of vowels in a given string.

Step-by-Step Explanation

1. Identify vowels: `a, e, i, o, u`.
2. Convert the string to lowercase so comparison is easier.
3. Loop through each character.
4. Check if character is in the vowel list.
5. If yes, increase count.

Final Function

```
function countVowels(str) {
  const vowels = "aeiou";
  let count = 0;

  // Loop through each character
  for (let char of str.toLowerCase()) {
    if (vowels.includes(char)) {
      count++;
    }
  }
```

```
  return count;
}
```

---

4. Largest Number in Array

Input

An array of numbers. Example: `[3, 10, 6, 8]`

Output

Largest number. Example: `10`

Problem Statement

Find the biggest number in an array.

Step-by-Step Explanation

1. Assume the first number is the largest.
2. Start comparing each number with this assumed maximum.
3. If any number is bigger, update max.
4. Continue until all numbers are checked.

Final Function

```
function findMax(arr) {
  let max = arr[0]; // assume

  for (let num of arr) {
    if (num > max) {
      max = num; // update
    }
  }

  return max;
}
```

---

5. Remove Duplicates from Array

Input

An array with duplicate values. Example: `[1,2,2,3,1]`

Output

Array without duplicates. Example: `[1,2,3]`

Problem Statement

Return a new array with all unique values.

Step-by-Step Explanation

1. Create an empty result array.
2. Loop through original array.
3. Before pushing a value, check if it already exists.
4. If not found, add it.
5. If found, skip.

Final Function

```
function removeDuplicates(arr) {
  let result = [];

  for (let num of arr) {
    if (!result.includes(num)) {
      result.push(num);
    }
  }

  return result;
}
```

---

6. Read a Nested Object Using Path

Input

Object and path string. Example object:

```
{
  student: {
    address: {
      city: "Hyderabad"
    }
  }
}
```

Path: `"student.address.city"`

Output

Value at the path. Example: `"Hyderabad"`

Problem Statement

Access values like `student.address.city` safely.

Step-by-Step Explanation

1. A path like `a.b.c` means nested keys.
2. Split the string using `.`, giving an array `["a", "b", "c"]`.
3. Start with main object.
4. For each key, go one level deeper.
5. Stop and return `undefined` if a key doesn't exist.

Final Function

```
function readNested(obj, path) {
  const keys = path.split(".");
  let current = obj;

  for (let key of keys) {
    if (current[key] === undefined) return undefined;
    current = current[key];
  }

  return current;
}
```

---

7. Character Occurrence Counter

Input

A string. Example: `"apple"`

Output

Object with counts. Example: `{ a:1, p:2, l:1, e:1 }`

Problem Statement

Count how many times each character appears.

Step-by-Step Explanation

1. Create an empty object.
2. Loop through each character.
3. If character already exists in object → increase count.
4. If not, create and set count to 1.

Final Function

```
function charCount(str) {
  let result = {};

  for (let char of str) {
    if (result[char]) {
      result[char]++;
    } else {
      result[char] = 1;
    }
  }

  return result;
}
```

---

8. Sorting Array Without sort()

Input

Array of numbers. Example: `[5,2,8,1]`

Output

Sorted array. Example: `[1,2,5,8]` ()

Problem Statement

Sort numbers from smallest to largest.

Step-by-Step Explanation

1. Use two loops.
2. Outer loop selects a number.
3. Inner loop compares it with the rest.
4. If a smaller one is found → swap.

Final Function

```
function sortArr(arr) {
  for (let i = 0; i < arr.length; i++) {
    for (let j = i + 1; j < arr.length; j++) {
      if (arr[j] < arr[i]) {
        let temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
      }
    }
  }
  return arr;
}
```

---

9. Flatten a Nested Array

Input

Nested array. Example: `[1,[2,3],[4,[5]]]`

Output

Flattened array. Example: `[1,2,3,4,5]`

Problem Statement

Convert nested arrays like `[1, [2, 3], 4]` into `[1, 2, 3, 4]`.

Step-by-Step Explanation

1. Create empty result array.
2. Loop through each item.
3. If item is not array → push directly.
4. If item is array → flatten it first, then push values.

Final Function

```
function flatten(arr) {
  let result = [];

  for (let item of arr) {
    if (Array.isArray(item)) {
      let flat = flatten(item);
      for (let f of flat) result.push(f);
    } else {
      result.push(item);
    }
  }

  return result;
}
```

---

10. Average of Array

Input

Array of numbers. Example: `[10,20,30]`

Output

Average. Example: `20`

Problem Statement

Find average value of numbers.

Step-by-Step Explanation

1. Add all numbers.
2. Divide sum by count.

Final Function

```
function average(arr) {
  let total = 0;
  for (let n of arr) {
    total += n;
  }
  return total / arr.length;
}
```

---

11. Find All Strings in Nested Object

Problem Statement

Extract all string values from deeply nested objects.

Input

```
{
  a: "hello",
  b: { c: "world", d: { e: 10, f: "js" } }
}
```

Output

`["hello", "world", "js"]`

Step-by-Step Explanation

1. Create empty result array.
2. Loop through object keys.
3. If value is string → store it.
4. If value is object → call same function again.
5. Combine all string results.

Function

```
function findStrings(obj, result = []) {
  for (let key in obj) {
    const value = obj[key];
    if (typeof value === "string") {
      result.push(value);
    } else if (typeof value === "object") {
      findStrings(value, result);
    }
  }
  return result;
}
```

---

12. Palindrome Finder in a Sentence

Problem Statement

Given a sentence, find: - How many palindrome words exist - The longest palindrome

Input

`"madam speaks level words in noon time"`

Output

```
Count: 3
Longest: "madam"
```

Step-by-Step Explanation

1. A palindrome reads same forward and backward.
2. Split sentence into words.
3. For each word, reverse it manually.
4. If original === reversed → it's a palindrome.
5. Count them and track longest.

Function

```
function palindromeAnalysis(sentence) {
  const words = sentence.split(" ");
  let count = 0;
  let longest = "";
```

```
  function reverseWord(w) {
    let r = "";
    for (let i = w.length - 1; i >= 0; i--) r += w[i];
    return r;
  }

  for (let w of words) {
    let reversed = reverseWord(w);
    if (w === reversed) {
      count++;
      if (w.length > longest.length) longest = w;
    }
  }

  return { count, longest };
}
```

(More programs can be added on request.) if you want 10 more problems, advanced tasks, or real company-level assignments.)*