🔤 String Interview Quick-Check Pattern

Your 1-minute pre-interview scan to conquer string-based problems confidently.

---

1. 🧠 What's the Nature of the String Problem?

- ✅ Is it about **pattern**? (Palindrome, anagram, duplicates)
- ✅ Is it about **matching/searching**? (Substring, indexOf, contains)
- ✅ Is it about **transforming**? (Reverse, case change, decode)
- ✅ Is it about **generation**? (Permutations, recursion, subsets)
- ✅ Is it about **optimization**? (Longest/shortest substring, window)

> 💡 Classify the problem type fast — this directs your thinking immediately.

---

2. 🔍 Most Useful Techniques

Technique	Used For
Hash Map / Set	Frequency, anagrams, duplicate checks
Two Pointers	Palindromes, trimming, swaps
Sliding Window	Longest/shortest substrings, window min/max
Stack	Balanced parentheses, decode strings
Recursion/Backtracking	Permutations, subsets, restore IPs
Dynamic Programming	Edit Distance, Longest Common Subsequence
Regex	Clean or extract data

> 🔁 You'll see these 7 patterns over and over. Lock them in.

---

3. 🧭 Understand What’s Being Asked

- ❓ Are you returning a **string**, a **boolean**, a **length**, or **an index**?
- ❓ Are you counting something, building something, or checking a condition?
- ❓ Are there **multiple strings** to compare (e.g. anagram check)?
- ❓ Are you modifying input, or generating output?

---

4. 🧪 Core Templates & Patterns

✅ Palindrome Check (cleaned)

```
function isPalindrome(s) {
  s = s.toLowerCase().replace(/[^a-z0-9]/g, '');
  let l = 0, r = s.length - 1;
  while (l < r) {
    if (s[l++] !== s[r--]) return false;
  }
  return true;
}
```

✅ Longest Substring Without Repeating Characters

```
function lengthOfLongestSubstring(s) {
  let set = new Set(), l = 0, max = 0;
  for (let r = 0; r < s.length; r++) {
    while (set.has(s[r])) {
      set.delete(s[l]);
      l++;
    }
    set.add(s[r]);
    max = Math.max(max, r - l + 1);
  }
  return max;
}
```

✅ Group Anagrams

```
function groupAnagrams(strs) {
  const map = {};
  for (let word of strs) {
    const key = word.split('').sort().join('');
    if (!map[key]) map[key] = [];
    map[key].push(word);
  }
  return Object.values(map);
}
```

✅ Minimum Window Substring

```
function minWindow(s, t) {
  const need = {}, window = {};
  for (let c of t) need[c] = (need[c] || 0) + 1;

  let have = 0, needCount = Object.keys(need).length;
  let l = 0, res = [-1, -1], len = Infinity;

  for (let r = 0; r < s.length; r++) {
    let c = s[r];
    window[c] = (window[c] || 0) + 1;
    if (window[c] === need[c]) have++;

    while (have === needCount) {
      if (r - l + 1 < len) {
        res = [l, r];
        len = r - l + 1;
      }
      window[s[l]]--;
      if (window[s[l]] < need[s[l]]) have--;
      l++;
    }
  }

  return len === Infinity ? '' : s.slice(res[0], res[1] + 1);
}
```

✅ All Permutations of a String

```
function permute(str) {
  const res = [];
  const backtrack = (path, used) => {
    if (path.length === str.length) {
      res.push(path.join(''));
      return;
    }
    for (let i = 0; i < str.length; i++) {
      if (used[i]) continue;
      used[i] = true;
      path.push(str[i]);
      backtrack(path, used);
      path.pop();
      used[i] = false;
    }
  };
  backtrack([], []);
  return res;
}
```

---

5. 🧱 Edge Cases to Watch For

- Empty string
- Case sensitivity (e.g. `"A"` vs `"a"`)
- Punctuation and special characters

- Whitespace
- Single character
- Duplicates
- Index out of bounds (e.g. substring slicing)
- Unicode (in advanced cases)

🧠 Think: What would **break** this logic with corner input?

---

6. 🧠 Mental Model to Store Forever

🧩 All string problems fall into these 5 categories:

Category	Trigger Word Examples	Key Techniques
Check	isPalindrome, isAnagram	Two Pointers, HashMap
Search	indexOf, minWindow, contains	Sliding Window, Frequency Counter
Transform	reverse, compress, decode	Stack, Two Pointers
Generate	all substrings, permutations	Backtracking, Recursion
Compare	longest common subsequence	Dynamic Programming

🔁 Problem Solving Loop:

1. 🔍 Identify which of the 5 string types this is.
2. 🛠️ Pick the right technique from the toolkit.
3. 🧱 Plug in templates.
4. 🚨 Watch for edge cases.
5. 🧼 Clean the string (lowercase, remove punctuation) if needed.

✅ Final Interview Checklist

- What is the **goal**? (Check, Search, Transform, Generate, Compare)
- What is the **output**? (String? Boolean? Integer?)
- Is it **optimized** for time and space?
- Did I handle **edge cases**?
- Can I apply a **known pattern** (sliding window, stack, backtrack)?
