★ Count words :

① "This | is | a | sample | string"

arr = ["This", "is", "a", "sample", "string"] ⇒ str.split(" ");

⇒ arr.length ⇒ (5)

② "This | | is | | | a | sample | string"

arr = ["This", "", "is", "", "", "a", "sample", "string"] ⇒ str.split(" ");

(1) (2) (3) (4) (5)

⇒ arr.length ⇒ 8 ✗

(5) words

★ even after spliting,
simply iterate over array &
count non-empty strings.

★ Toggle characters :

S = "hello ABC"

"HELLO abc"

ans = "
= "H"
= "HE"
= "HEL"
= "HELLO abc"

1. go to every character check, whether it is upper/lower
2. If it is upper, add lower to ans
3. if it is lower, add upper to ans

★ How to check whether the ch is upper/lower ?

upper → 'A' <= ch <= 'Z' ⇒ if ('A' <= ch && ch <= 'Z')

lower ↘ 'a' <= ch <= 'z' ⇒ if ('a' <= ch && ch <= 'z')

★ pan grams :

'a' - 'z'

We promptly judged antique ivory buckles for the next prize → given string

should contain all the alphabets (case doesn't matter)

('A b C d e F g H I j K L m n o P Q R s t U V W X y Z')

→ pan gram

1. As it is case-insensitive, convert whole str to either lower / upper case. ✓

2. check whether
   - 'a' exists or not ? ⇒ str.includes('a')
   - 'b' exists or not
   - 'c' exists or not
   - ⋮
   - 'y' exists or not
   - 'z' exists or not

★ 'a' → 'z' iterate from 97 → 122 and you can use fromCharCode !

★ keep on checking until you find an alphabet which is not in the string until str.includes(ch) ⇒ false

Ascii values :

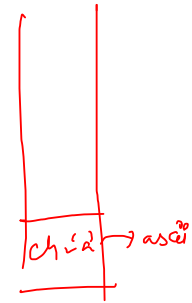
every character has a integer number associated with it

* 'a' → 'z' 97 → 122

a - 97
b - 98
c - 99
d - 100
⋮
y - 121
z - 122

* 'A' → 'Z' 65 → 90

'A' - 65
'B' - 66
'C' - 67
⋮
'Y' - 89
'Z' - 90



* ch → ascii ⇒ ch.charCodeAt(0)

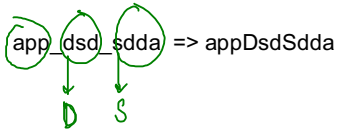
ascii → ch ⇒ String.fromCharCode(ascii)

"ABC".charCodeAt(1) ⇒ 66

String.fromCharCode(65) ⇒ "A"

"A".charCodeAt(0) ⇒ 65

✱ Camel case :



1. extract each word → str.split("_")
2. apart from 1st word, capitalize 1st letter of each word.

app | dsd | sdda

["app", "dsd", "sdda"] ⇒ as .join("")

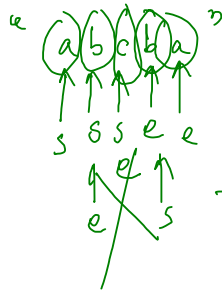
0 1 2 ⇒ appDsdSdda

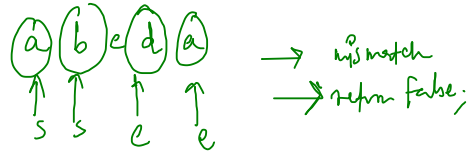
★ palindrome string :

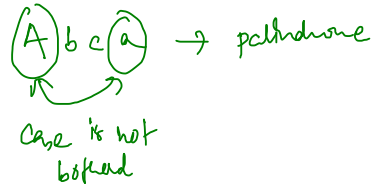
⇒ "a man, a plan, a canal Panama"

⇓ (case-in-sensitive)

"amanaplanacanalPanama"







- When comparing when you see a character which is not an alphabet skip it.

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28

a man, a plan, a canal Panama

e s

"a" "m" → s++

e, "a" → s++

a, "a" → s++

a, "p" → s++

"l", e → e--

a, "a" → s++

a, "a" → s++

a, "c"
→ s++

★ It is same as old palindrome, but only skip is extra

```
if (s[start] < 'a' || s[start] > 'z')
    if (e < 'a' || e > 'z') {
        start++
    }
```

★ Compressed Strings :

"a a a b b c c d d d a a" ⇒ "a3 b2 c2 d3 a2"

a3 b2 c2 d3 a2

last exploration was missed so add it

cnt = 1 2 3 1 2 1 2 1 2 3 1 2

1. at every char, check prev char if same cnt ++

if not same

→ prev_char's count is cnt.

ans += prev_char + count

cnt = 1

ans = "a"

= "a3"

= "a3b2"

= "a3b2c2"

= "a3b2c2d3"

= "a3b2c2d3a2"

"a a a b b b a a b b"

⇒ "a3 b3 a2 b2"

"a b b d c c"

⇒ ~~"a1 b2 d1 c2"~~ ⇒ "a b2 d c2"