evaluatable

eval func^n → it is capable of executing any expression

passed in it.

string

g ↜ eval("4+5")

$x = 10$

$y = 20$

eval(`${x} = ${y}`) → $x = y$

Template Strings

{
  name
  price
  brand
}

$\rightarrow$

The name of the product is ———— which

belongs to ———— brand with a price ————

all the strings
that starts with
'so'

Regex

↳ Regular Expression

pattern that helps to group
similar kind of strings
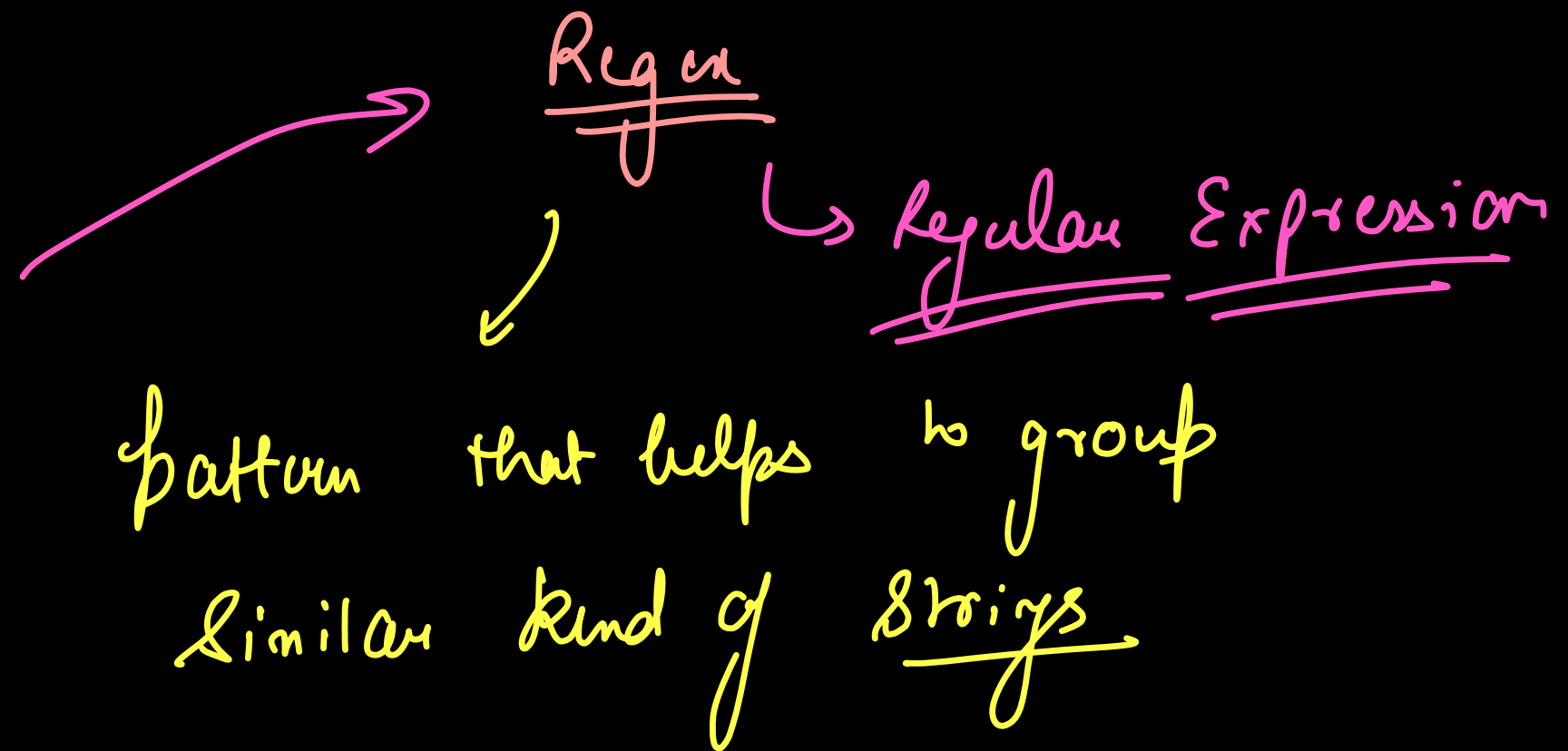
all string
that starts
with

So_____y
          ↓
1 or more no. g
          y

Sory
Sorry
Sorrrry
:
:
:
:

Components

① Character Classes ⇒ Set of characters we are going to allow

in the strings

$[0-9] \longrightarrow$ any digit

$[a-z] \longrightarrow$ any small alphabet

$[A-Z] \longrightarrow$ any cap alphabet

$[980]$

$\boxed{g} \rightarrow$ global search $\rightarrow$ find all the possible chars that follows the regex.

$$/ \underline{\qquad} /g$$

↓

if no g is given then only the first match is <u>returned</u>

$\boxed{i} \rightarrow$ this means a case insensitive search

$$/ \underline{\qquad} /g^i$$

$$/ \underline{\qquad} /i$$

Quantifiers → It is used to check how many times a char should occur.

→ * → means 0 or more occurred of any char

So *y

+ → mean 1 or more occurred of a char

`/[a-zA-Z0-9_.]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}/`

^
$

Start with one
or more occ of
a-z A-Z 0-9 _ .

followed by
@

one or more
occ of a-zA-Z
0-9 . -

followed by
one .

2-4
lphu occ
of
a-z A-Z

application

→ ① authentication → match valid email
  stray password

② frontend → form validation ✓

③ Compiler & interpreter