Regular expressions

→ It enables finding patterns in a string.

→ Use case: Text validation
↳ Search through string (find no.)

Eg 1) Find 'helloworld' in a string.

```
Var helloworld = document.getElementsByTagName("simplehello");
Const text = "People often write hello world while programming";
Const regex = /hello world/;
Const res = text.match(regex);
helloworld.innerHTML = res;
console.log(res);
```

o/p
['hello world', index: 19, input: 'People ___', groups: undefined]

Eg 2) findmany → Global option or Modifier → /g

```
Const text2 = "People often write hello world while hello world programming";
Const regex2 = /hello world/g;
Const res2 = text2.match(regex2);
console.log(res2);
```

Same o/p

This o/p gives you only 1st match

If you want many matches, you use below:

```
Const regex3 = /hello world/g;
```

use this

```
Const res2 = text2.matchAll(regex2)
```

The o/p will be

['helloworld', index: 19, input: '...', groups:]
['helloworld', index: 37, input: '...']

Eg 3) Case Sensitive (i) option → For case sensitive

```
const text4 = "People often write hello world while HELLO WORLD
               programming";
const regex4 = /hello world/gi;
const results2 = [...text4.matchAll(regex4)];
results2.forEach(res => {
    console.log(res);
});
```

o/p

['helloworld', index: 19, input: 'c_' groups: undefined]
['helloworld', index: 35, input: 'c_' groups: undefined]

Eg 4) Find Capital letter

```
const text5 = "People often write hello world HELLO WORLD
               programming";
const regex5 = /[A-Z]/g;
const results3 = text5.match(regex5);
console.log(results3);
```

o/p

(12) ['P', 'W', 'H', 'E', 'L', 'L', 'O', 'W', 'O', 'R', 'L', 'D']

Eg 5) Find Capital letter & Ignore letter

RE 1

```
const regex6 = /[A-Og-z]/g;
```

replace with this for eg 5

doubt ?

OR

RE 2

```
const regex5 = /(?:[P])[A-Z]/g;
```

Both will give same o/p → (11) ['W', 'H', 'E', 'L', 'L', 'O', 'W', 'O', 'R', 'L', 'D']

Eg 6) Find Digits

Why do we happen to put /g option to find no.? → We get full array.
if we do not?

```
const txt4 = "Around 20 years ago, there were 5 boys and 6 girls who lived in a village.";

const regex4 = /\d/g ;

const numFind = txt4.match(regex4);
console.log(numFind);
```

o/p : (4) ['2', '0', '5', '6']

Replace with this! →

```
const regex4 = /\d+/g ;
```

↳ To get full numbers.

You get o/p → (3) ['20', '5', '6']

Eg 7) Replace → Use 't' operator

```
const txt4 = "Color is Black & code is 8 ";

const regex4 = /\d+/g ;
const numFind = txt4.replace(regex4, "9");
console.log(numFind);
```

and

o/p : Color is Black & code is 9

Eg 8) Search() → Give us position of first match

```
const text = "It is a huge big big place of a big world";
const regex = /big/g;
const numFind = text.search(regex);
console.log(numFind); //13
```

o/p : 13

Eg 9) Exec() → Similar to match()

→ But purpose used in a loop.

↳ returns array of captures or null.

↳ In Regular expression, lastIndex property used with exec() to determine index which next match will start.

↳ The property is updated whenever a match is found with the g flag.

↳ regex.lastIndex → Auto updated Not manually modify

↳ After loop finish → regex.lastIndex → reset to 0

```
const text = "Big and large world world full of games";
const regex = /world/g
let arr;
while (arr = regex.exec(text) != null) {
  console.log(`Found ${arr[0]}. Next starts at ${regex.lastIndex}.`);
}
```

o/p
Found world. Next starts at 19.
Found world Next starts at 26.

match()	exec()
→ Method of string	Method of Regular expression
→ If match found returns arrays containing matched text as 1st element & add elements represent any captured group if present	[ ]

Eg 10) Email validation

/^[^\s@]+@[^\s@]+\.[^\s@]+$/g

- ^ → Assert Start of String
- [^\s@]+ → Matches one or more char that not whitespace or @ before @
- @ → After @
- [^\s@]+ → Matches Any
- \. → After Dot
- [^\s@]+ → Matches Any
- $ → Assert End of String

\s → Whitespace character

\S → Non whitespace character

[ ] → Specify a set of char that you want to match

\. → Math • symbol (Escaping because it has special meaning in regex)

+ → quantifier → matches preceding element one or more times

\d → Digit

```
function validateEmail() {
  let giveemail = document.getElementById("email").value;
  let printRes = document.getElementById("result");
  let email = giveemail;
  const emailExp = /^[^\s@]+@[^\s@]+\.[^\s@]+$/g;
  const res = email.match(emailExp);
  if(res) {
    printRes.innerHTML = "valid email";
  } else {
  }
}
```