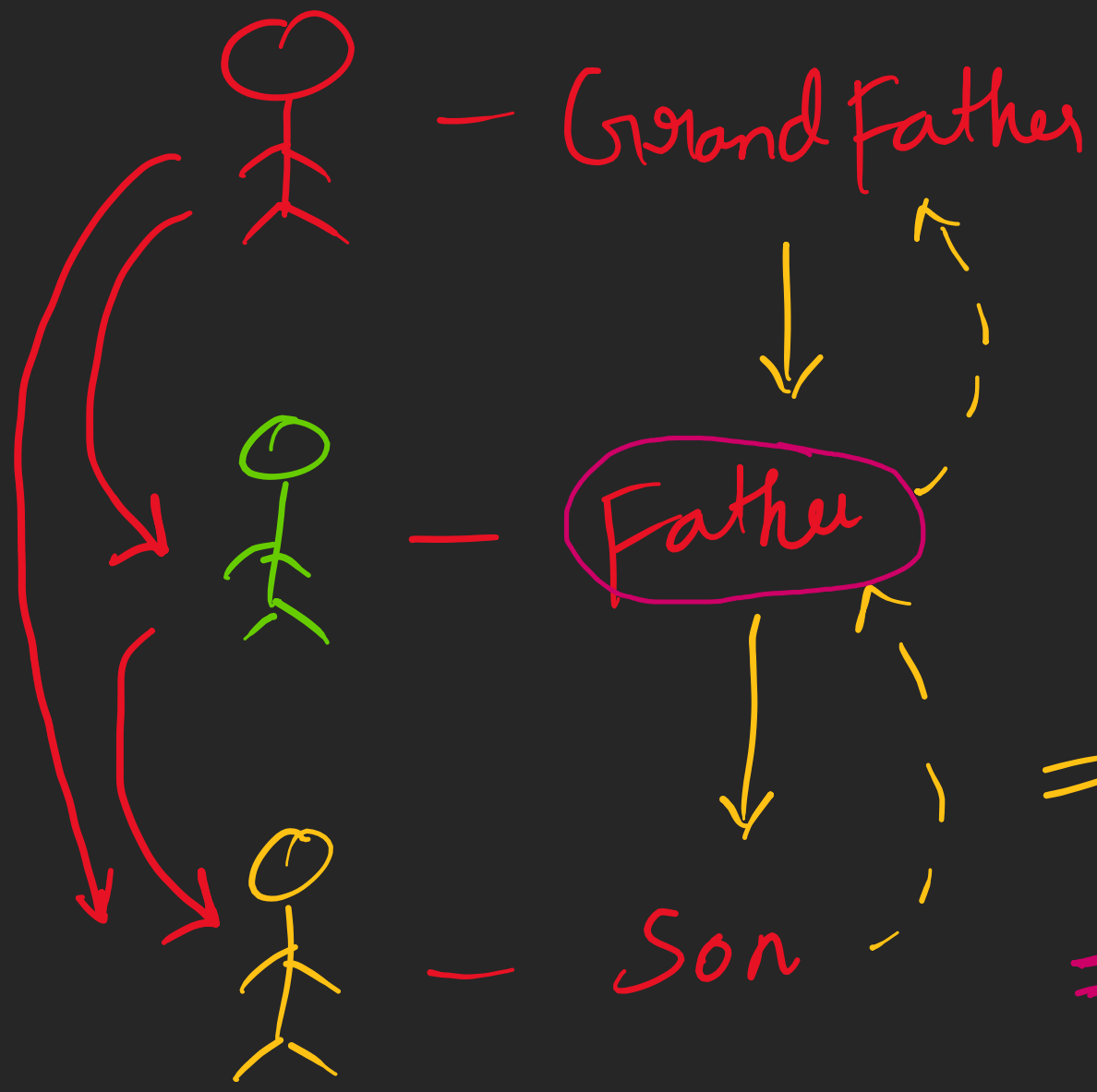
Day - 7

- → classes and objects [inheritance/prototype]
- → super() keyword
- → this keyword
- → Map, reduce, filter
- → setter & getter (class, function (return), obj)



[. - - - pseudo - - -]

* Inheritance (variables, functions)

⇒ Father extends Grandfather
(Father + Grand Father)

⇒ Son extends Father

Son + Father + Grand Father

'this' keyword :-

* In JavaScript, the 'this' keyword is a special "identifier" that represents the current execution context or object within a given function or code block.

* Its behaviour is determined by the context in which it is used, and it can refer to different objects or values depending on that context.

Arrow Functions:

→ The 'this' keyword inside the outer function points to the "window object"

→ Now, when the inner function uses the 'this' keyword, it actually references the scope of the outer function.

→ Since the inner function gets its 'this' reference from the outer function which now points to the global object, the inner function also starts pointing to the global object

Object:

* function() → function()
  (points to obj) (points to windows obj)

* function() → () => { } arrow
  (points to obj) (points to obj(function()))

* () => { } → () => { }
  (points to window obj) (points to windows obj)

Same as above!

* () => { } → function()

Getter Methods:

→ Used to retrieve the value

→ define 'getter' method using the 'get' keyword!

→ Inside getter method, you can define custom logic to compute/format the value before it is returned.

Setter Methods:

→ Used to set the value. "Set" keyword.

→ Inside setter method, you can include validation logic to ensure the new value meets certain criteria.

literals and identifiers (fundamental concepts)

→ let name = "Guvi";

(variable name) (Value)

String [string literal]

→ let age = 30; [number literal]

→ let isStudent = true; [Boolean Literal]

"Variable identifier"

Small

↳ 15k/month [2 years profile 45-60k] + 1 year

knowledge 1.5

Medium

↳ 20k/month (18k) [2 years 28-30k]

Top

↳ 25-28k/month 4/5.52 PA [2 years 33-35k]

80k → 35k

20000
4000
24000
4800
28800

60
30%

28000
2800
30800
3080
33880

1. Trainee (fresher)
2. S.E (1-3 yr)
3. S.S.E (2.5 - 5 yr)
4. TA (3.5 - 6.5 yr)
5. TL (5.5 - 9 yr)
6. M

* Skill based
* previous company salary

20 yr = 40 LPA

India IT HR policy

Skill, technology, demand

1 yr = 2 LPA (min)
5 yr = 10 LPA

Monalisa Art → $ → 1M → 1B

Original Art

Web 2.0

Web 3.0 * PubG

fund → 100 → 100K

NFT

(NFT)

Web Application

"digital form" *

Non Fungible Token

System design

Name

=digital form

Author!

(Own hand written)

✓ Original copy (Value A)

✓ (photo copy - xerox) (Value B)

⇒ VR, Virtual Box

Vision pro

⇒ Blockchain

⇒ NFT

web3.0

AI

Machine Learning

⇒ Accenture

⇒ Infosys

Array Methods:

- → map( )
- → filter( )
- → reduce( )

Order food online :-

10% discount

(100 reviews)

average [4.5 rating]

1. Sangeetha [– , – , – , –]

2. SS Hyderabad [– , – , – , –] 5 rating

3. Junior kuppanna [– , – , –] 3 rating

4. KFC [– , – , – , –]

5. Mcdonald's [– , – , – , –]

[100, 3, 4, 5, 4.5, 3.2, 3.1]

= "4.5"

Chicken biriyani = 200 Rs ⇒ 180 Rs

"Buy 1 get 1 free"

0 = 0

0+1 = 1 finger ✗

1+1 = 2 finger ✗

2+1 = 3 finger ✗

3+1 = 4 finger ✗

1. Jar of candies

2. He uses hand to count

3. Pick one candy and add it to your hand

4. Always start with your empty hand (0)

(sys)

(empty => 0)

(hand)

(accumlator, currentValue)

```
map():
    function(item) {
        ____ ____
    }

filter():
    function(item) {
        ____ ____
    }
```

Reduce:

```
function (accumlator, currentValue) {

    - - - - -

}
```

```
let obj = {name: 'Mani', age: 32};
obj[name] = "Mani";  obj[age] = 32;
```

fruits = ['apple', 'orange', 'banana', 'orange', 'banana']

count['apple'] = null ← count = {};

fruits.reduce((count, fruit) => {

{apple:1}

count = {apple:1}

count['apple'] = (count[fruit] || 0) + 1;

1

count['orange'] = 2

return count;

},{});

{apple:1, orange:1}

↓

{apple:1, orange:2, banana:2}