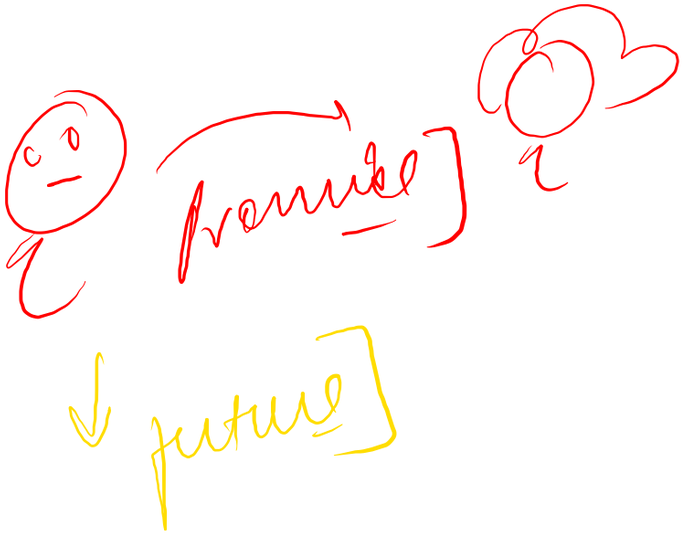
(JS)

1. Promises
2. Async Await
3. Error Handling

Tricky | Interview

promises



Promise → Futuristic

Accept

Reject

completed

(VS)

Pending

Promise object

(+)ve

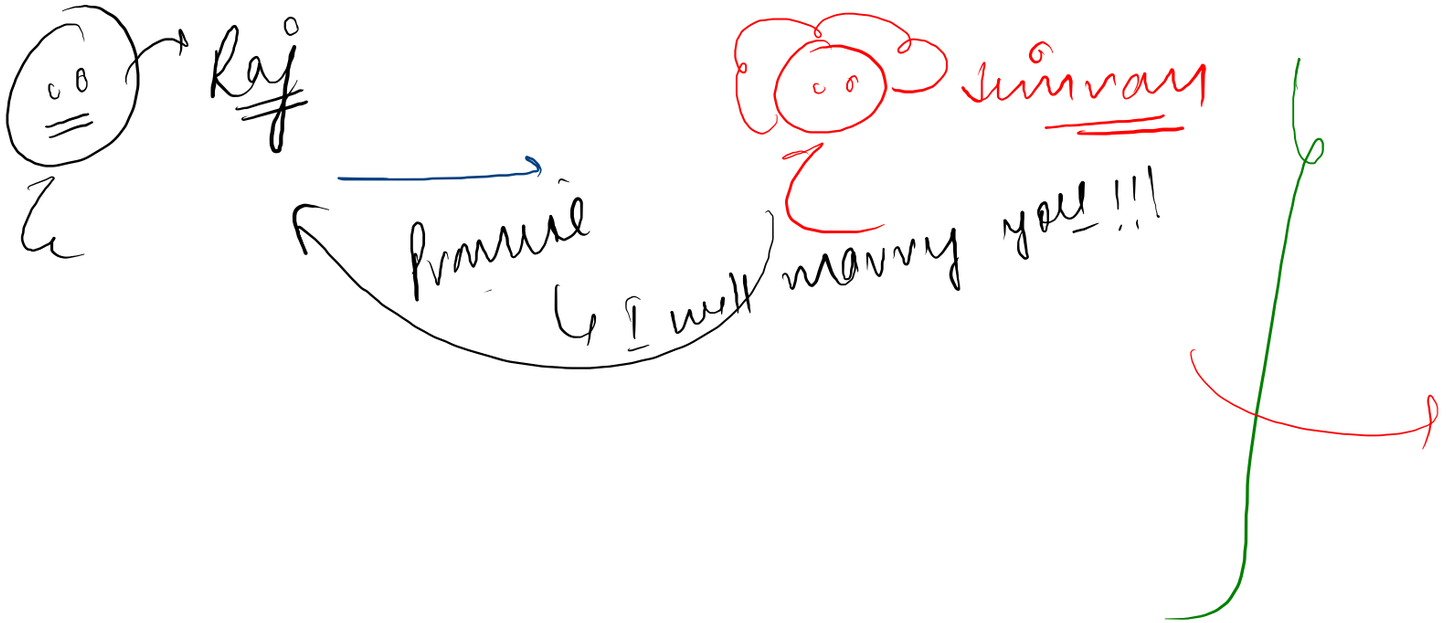
(-)ve outcome

Actor

Resolve

Reject

Settled / completed



(JS) → Promise object

object fn fn

[redacted]

```
const simranPromise = new Promise((resolve, reject)=>{

    let parentDecision = false

    if(parentDecision){
        resolve("Yayy, ab shaadi ki tayyari karo")
    }else{
        reject("Papa ne IAS dhoond lia hai !, Sorry ....")
    }
});

simranPromise.then((msg)=>{
    console.log("Simran message : ", msg)
    console.log("Let's book the wedding venue")
}).catch((msg)=>{
    console.log("Simran message : ", msg)
    console.log("Tinder kha ho yaar !")
})
```

Result

promise

(fn) → takes an argument → resolve

Negative

```
const simranPromise = new Promise((resolve, reject)=>{

    let parentDecision = false

    if(parentDecision){
        resolve("Yayy, ab shaadi ki tayyari karo", parentDecision)
    }else{
        reject("Papa ne IAS dhoond lia hai !, Sorry .... ", parentDecision)
    }

})

simranPromise.then((msg , pd)=>{
    console.log("Simran message : ", msg)
    console.log("Parent decision was ", pd)
    console.log("Let's book the wedding venue")
}).catch((msg,pd)=>{
    console.log("Simran message : ", msg)
    console.log("Parent decision was ", pd)
    console.log("Tinder kha ho yaar !")
})
```

Promise object

true → resolve

false → reject

function

fn

call back fn

2 arguments

fn

fn → resolve

execution

no decision

Promise object

Resolve()

Reject()

completion state

Pending state

Execution

Node JS / JS

Applications } → How and why we use Promise

(JS) → Single thread

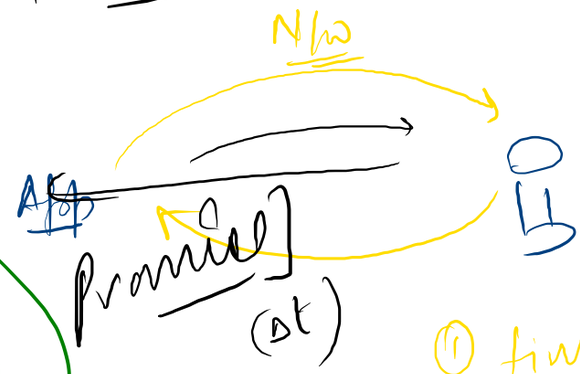
→ Asynchronous programming

① Read from database

② Process the data

dataPromise.then(↓ reject processing data)

.catch (to catch handle error)



(Δt) → ① data

① time
② processing } → ③ failure

Asynchronous processing

(S1) → (S2)

Promises

appropriate sequential tasks

Js file

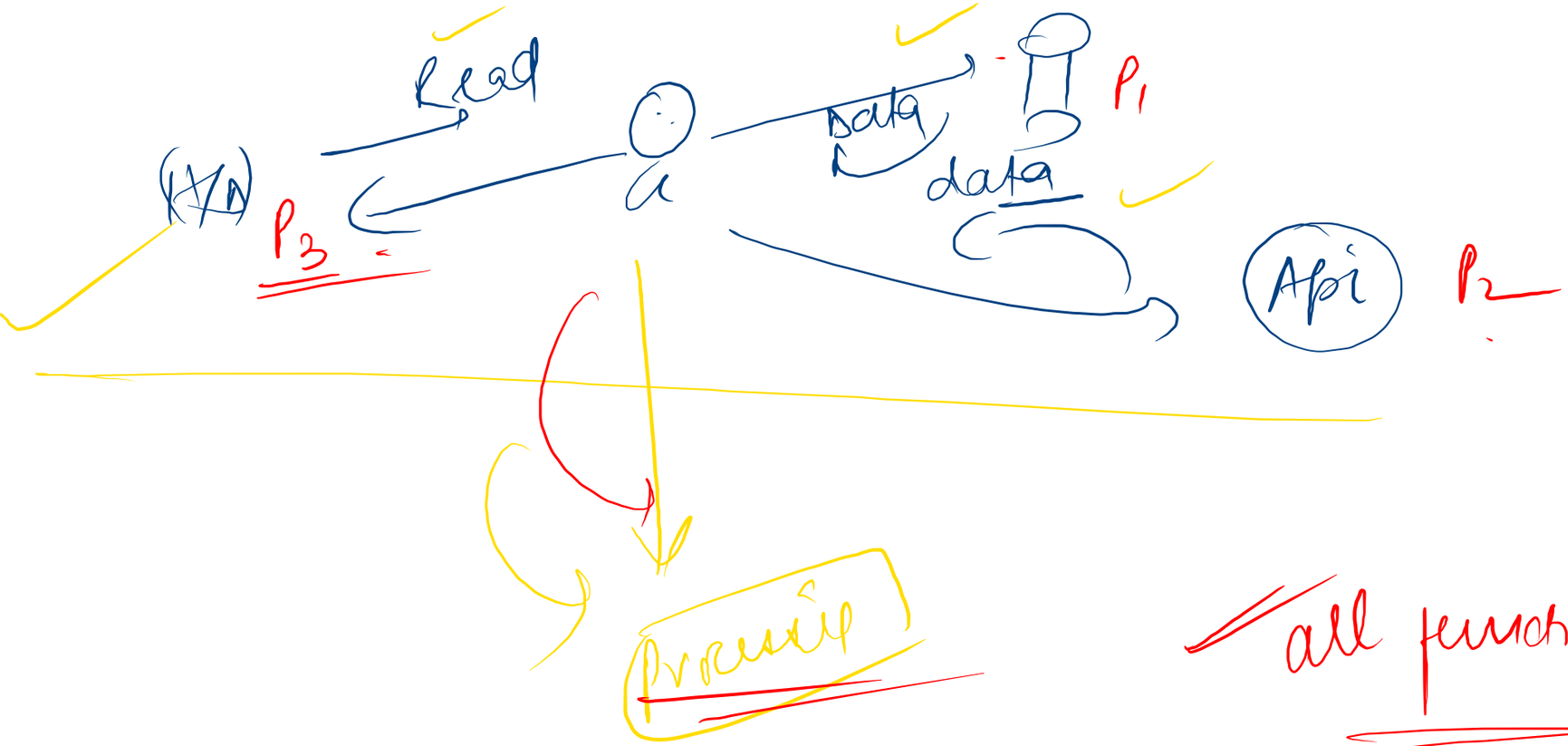
completed

P_1 → Async → (5-10 seconds)

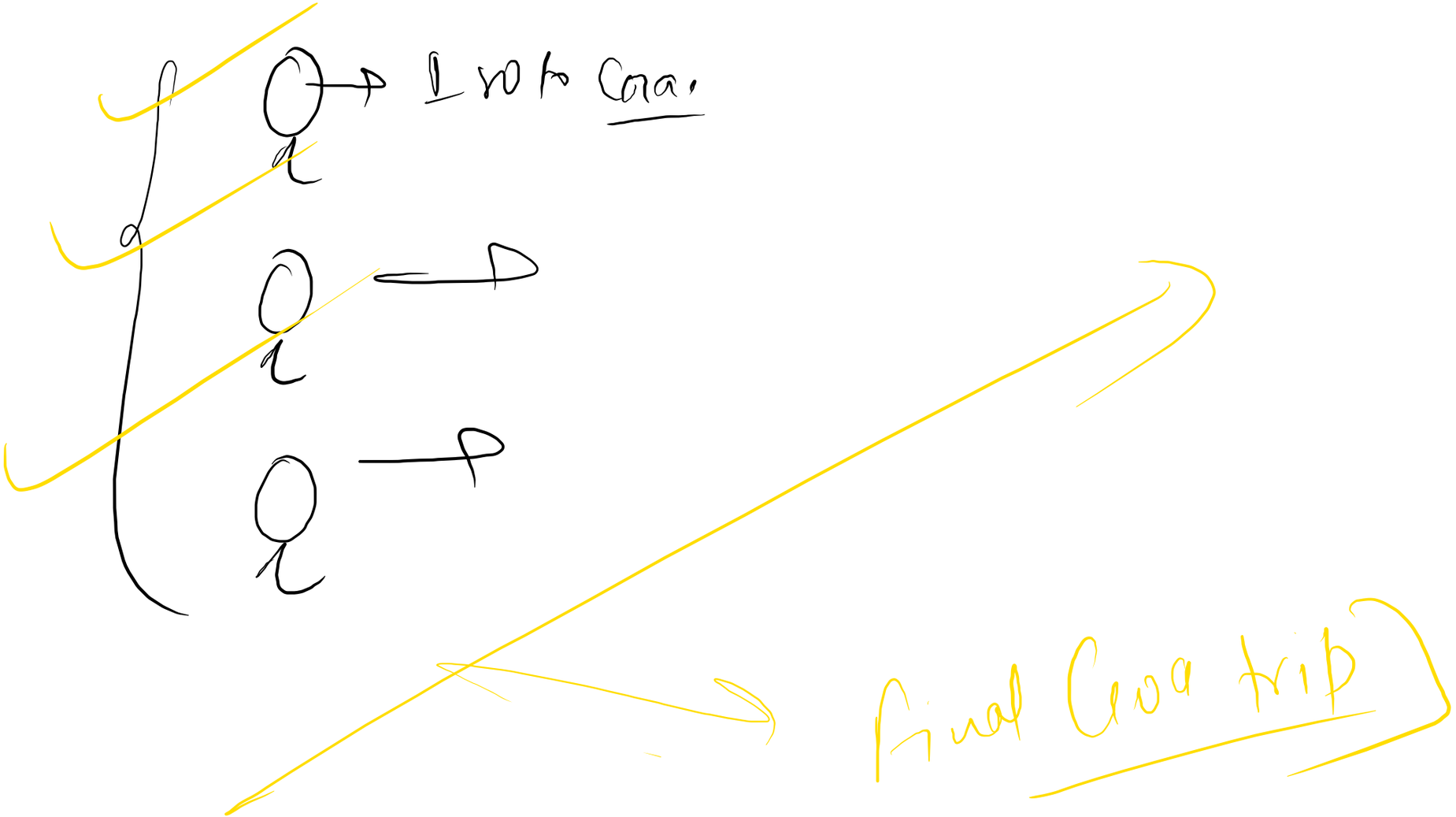
P_2 → Async → (2-12 seconds)

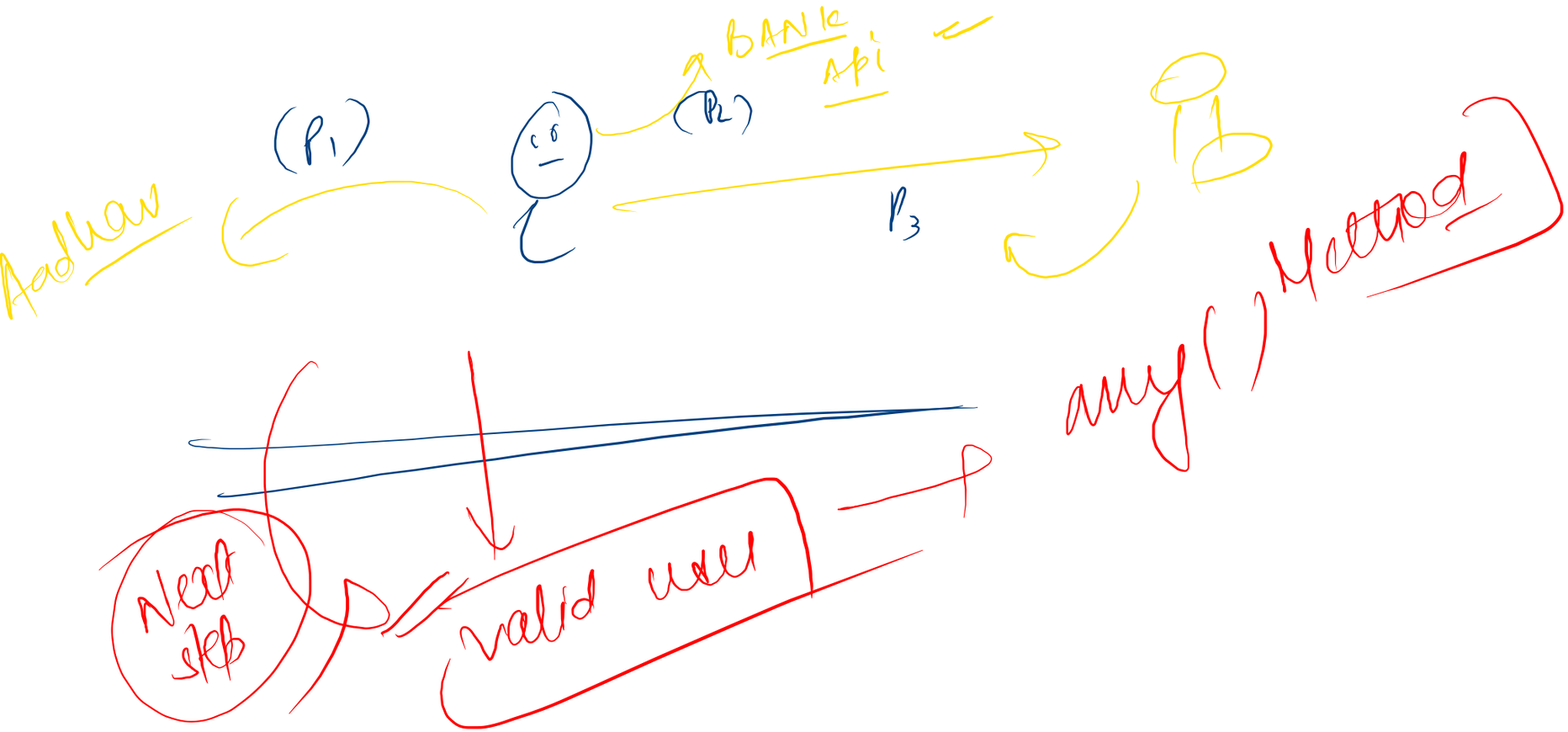
P_3 → Async → (16-20 seconds)

(S)



all function of promise


1 vote Goa.
final Goa trip


BANK API
(P1)
(P2)
P3
Aadhaar
any( ) Method
Next step
valid user

```
const f1Promise = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        if(Math.random() > 0.5){
            resolve("Friend1 : Hey I am in for Goa")
        }else{
            reject("Friend1 : Sorry mere pet mei dard ho rha hai !")
        }

    },3000)
})

const f2Promise = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        if(Math.random() > 0.5){
            resolve("Friend2 : Hey I am in for Goa")
        }else{
            reject("Friend2 :Sorry meri billi beemar hai !")
        }

    },3000)
})

const f3Promise = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        if(Math.random() > 0.5){
            resolve("Friend3 :Hey I am in for Goa")
        }else{
            reject("Friend3 :Sorry mujhe PW Skills ke live sessions karne ho !")
        }

    },1000)
})

Promise.all([f1Promise,f2Promise,f3Promise]).then((msg)=>{
    console.log(msg)
    console.log("Yay we are going goa !")
}).catch((msg)=> {
    console.log(msg)
    console.log("Yaar band karo Goa ka plan banana!")
})
```


```
const f1Promise = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        if(Math.random() > 0.5){
            resolve("Friend1 : Hey I am in for Goa")
        }else{
            reject("Friend1 : Sorry mere pet mei dard ho rha hai !")
        }

    },3000)
})

const f2Promise = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        if(Math.random() > 0.5){
            resolve("Friend2 : Hey I am in for Goa")
        }else{
            reject("Friend2 :Sorry meri billi beemar hai !")
        }

    },3000)
})

const f3Promise = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        if(Math.random() > 0.5){
            resolve("Friend3 :Hey I am in for Goa")
        }else{
            reject("Friend3 :Sorry mujhe PW Skills ke live sessions karne ho !")
        }

    },1000)
})
```

(f1)

```
const f2Promise = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        if(Math.random() > 0.5){
            resolve("Friend2 : Hey I am in for Goa")
        }else{
            reject("Friend2 :Sorry meri billi beemar hai !")
        }

    },3000)
})
```

(0,1)

time in ms

3 seconds

```
const f3Promise = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        if(Math.random() > 0.5){
            resolve("Friend3 :Hey I am in for Goa")
        }else{
            reject("Friend3 :Sorry mujhe PW Skills ke live sessions karne ho !")
        }

    },1000)
})
```

f1 Promise

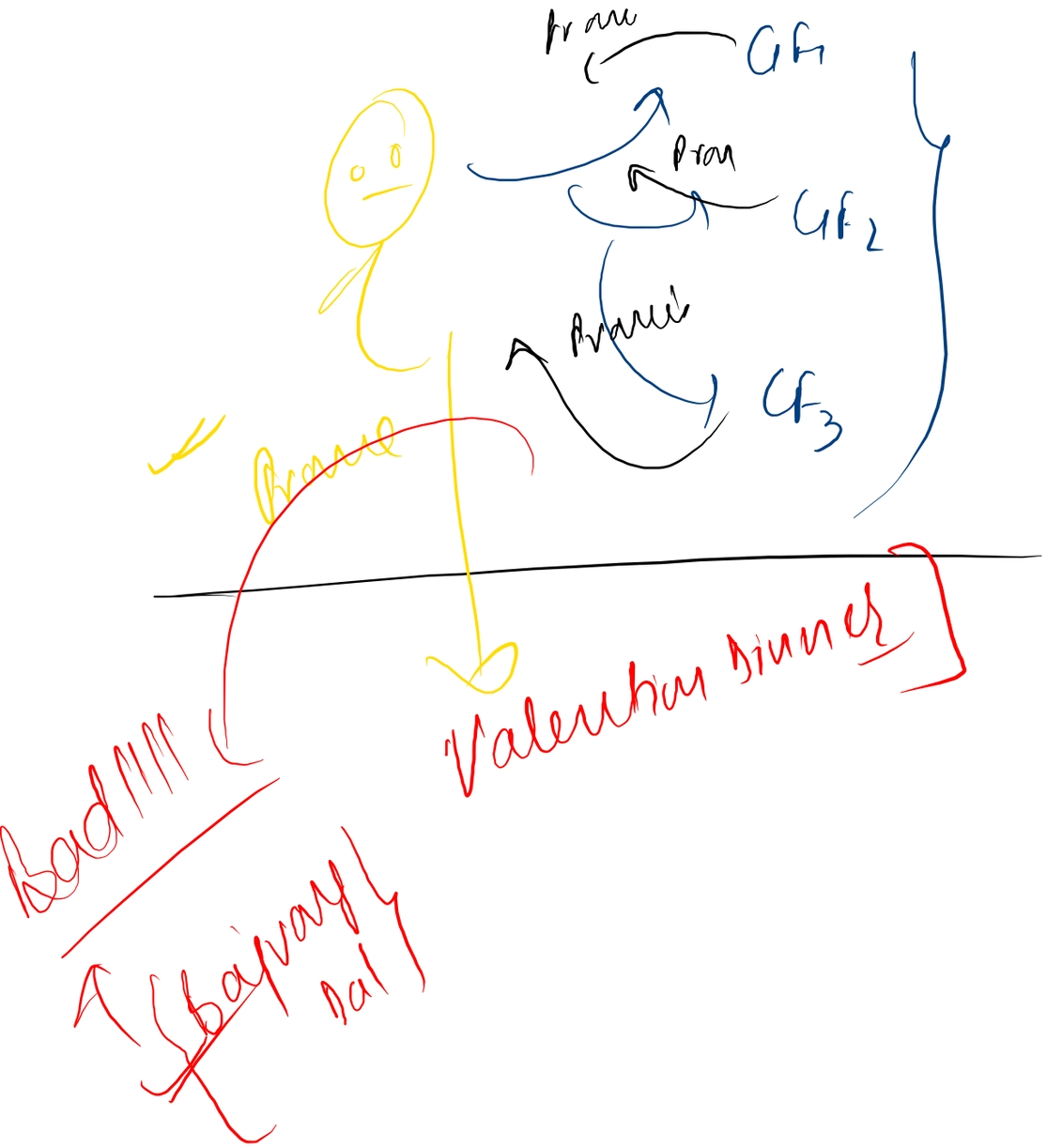
[Array of promises]

```
Promise.all([f1Promise,f2Promise,f3Promise]).then((msg)=>{
    console.log(msg)
    console.log("Yay we are going goa !")
}).catch((msg)=> {
    console.log(msg)
    console.log("Yaar band karo Goa ka plan banana!")
})
```

Promise any ( )

P_1 P_2 P_3

Next step


GF1
GF2
GF3
Valentine Dinner
Bad!!!!
Bajwa
Dal

```
const gf2Promise = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        if(Math.random() > 0.5){
            resolve("GF2 : Hey I'll come for dinner")
        }else{
            reject("GF2 : I am on diet !")
        }

    },1000)
})

const gf3Promise = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        if(Math.random() > 0.5){
            resolve("GF3 : Hey I'll come for dinner")
        }else{
            reject("GF3 : My firends are coming !")
        }

    },1000)
})
```

(0,1)

or condition

```
Promise.any([gf1Promise,gf2Promise,gf3Promise]).then(result=>{
    console.log(result)
    console.log("My valentine day is sorted")
}).catch(msg =>{
    console.log(msg)
    console.log("How to be the part of Bajrangdal ?")
})
```

if all positive

then

if any 1 rejects

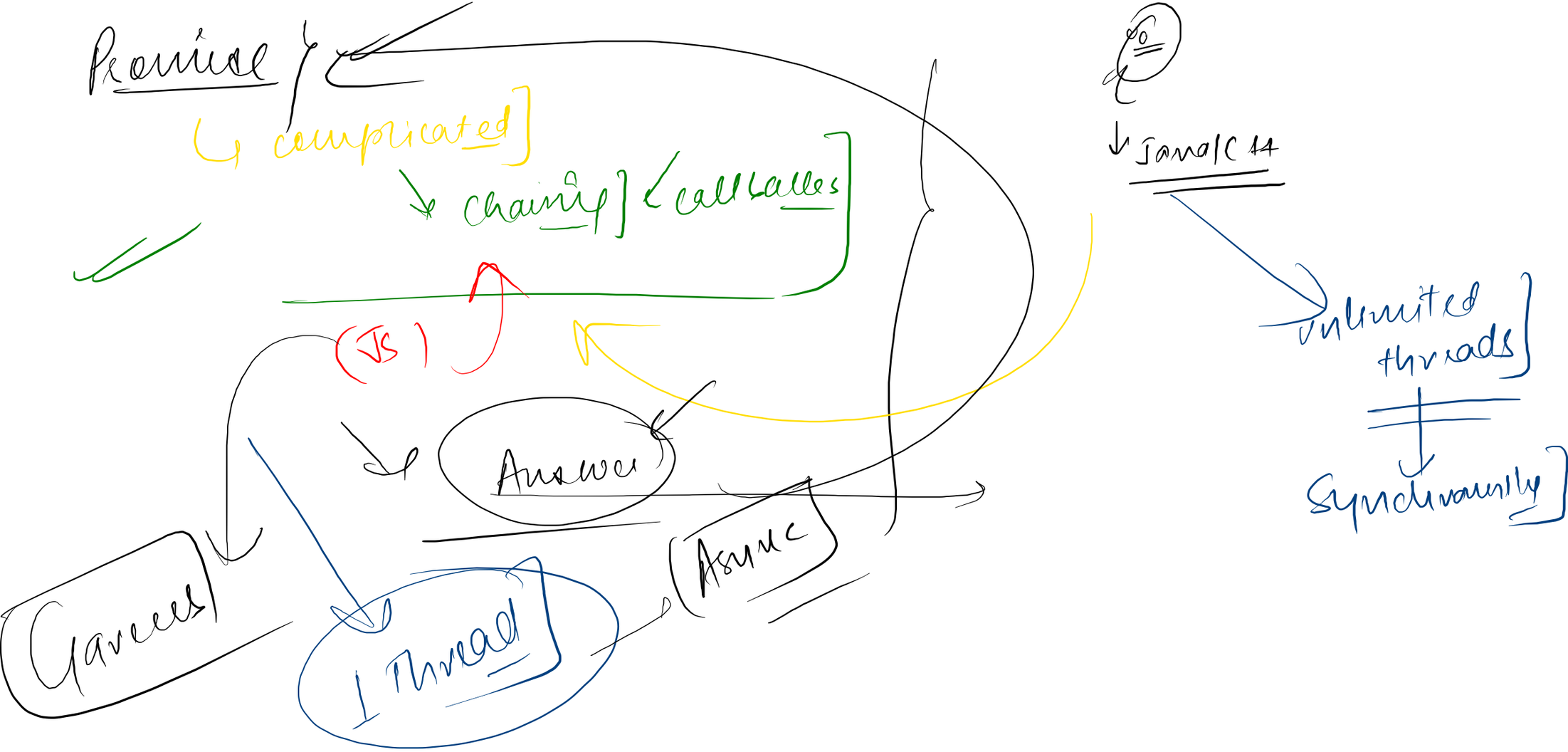
Promise.all ([P_1 P_2 , P_3]) → catch

if any selects & no

Promise.any ([P_1 P_2 P_3]) → then( ) → catch( )

any one positive

when all negative


Promise
complicated
chaining
callbacks
(JS)
Answer
Callbacks
1 Thread
Async
Java/C 14
Unlimited threads
Synchronously

JS/Node.js

Promise

complicated

Java/C#

Go

syntactical sugar

Async Await

it will se use of promise

Go

Java/C#

Async/Await

11:35

[Break time]

```
function printHelloAfterWait(){

    console.log('First Line')
    setTimeout(()=>{
        console.log("Hello Students")
    },3000)
    console.log('Last Line')
}


printHelloAfterWait()
```

Wait

wait

usecase

BE App

function process ( ) {

wait

code to fetch data

Process and return

error

Async Await

fn inside

waiting

Prefix

fn Async

Asynchronous / Non-blocking

```
console.log("Hello from begining ")
async function printHelloAfterWait(){

    console.log('First Line')

    let data = new Promise((resolve, reject)=>{
        setTimeout(()=>{
            resolve("Hello Students")
         },3000)
    })

    let result = await data     // Start waiting for the data promise to be completed
    console.log(result)
    console.log('Last Line')
}

printHelloAfterWait()

console.log("Hello from end")
```

waiting here

sequentially

waits for the promise to complete

Next step

Don't wait here

Hello from begining
First Line
Hello from end
Hello Students
Last Line

```
console.log("Hello from begining ")
async function printHelloAfterWait(){

    console.log('First Line')

    let data = new Promise((resolve, reject)=>{
        setTimeout(()=>{
            resolve("Hello Students")
         },3000)
    })

    let result = await data    // Start waiting for the data promise to be completed
    console.log(result)
    console.log('Last Line')
}

printHelloAfterWait()

console.log("Hello from end")
```

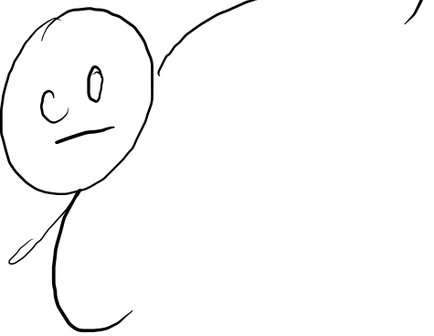
Hello from begining
First Line
Hello from end
Hello Students
Last Line

Async/Sync wait

to wait for the promise to resolve

Promises | Async Await

Error Handling

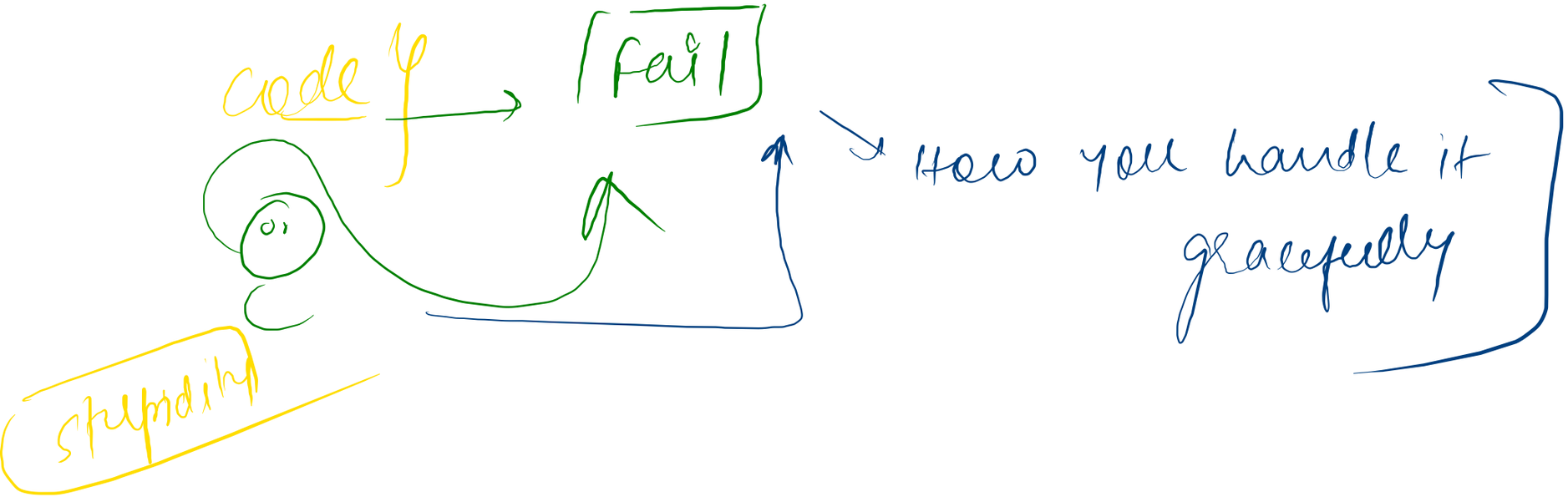


code

it will never fail!

Perfect code

Stupid x


Code
Fail
How you handle it gracefully
stupidity

Language → Handle exceptions

Error / Exception

Power is off

OOM

Happens

can't recover

App crash

Handled programmatically

try

catch

finally

```
try {
    ____
    ____
    ____
} catch(e) {
    ____ logic to handle it
}
```

[redacted]

```
try{
  let obj = undefined
  console.log(obj.name)
}catch(err){
    console.log("Exception handled")
}
```

something wrong

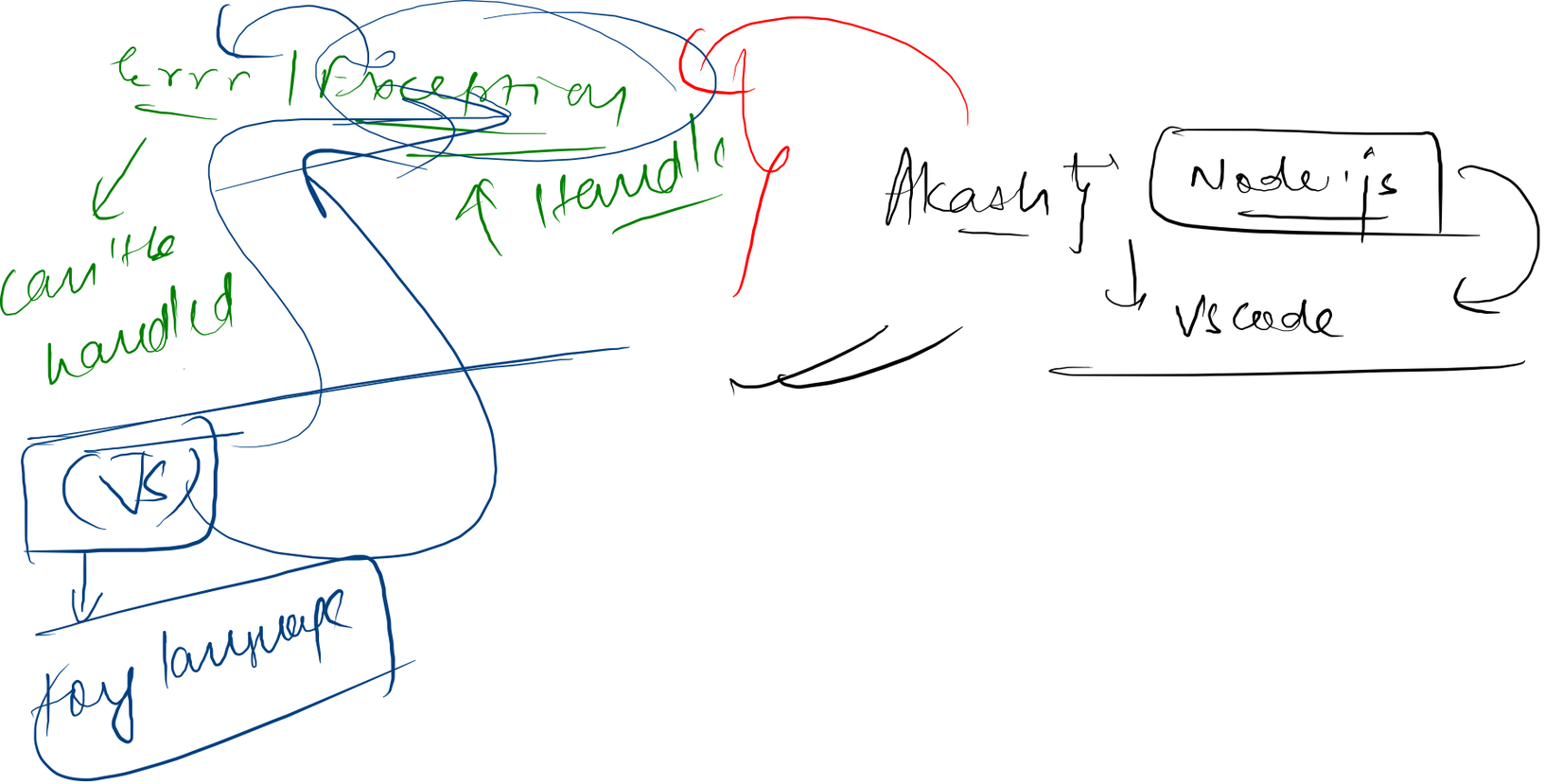
Promise . then ( ) . catch ( ) . finally ( )

(JS) | Node.js | MongoDB
↳ (ODM) → Mongoose | Express

App → Live

Economic
DE

3rd week


error | Exception
can't be handled
Handle
VS
Toy language
Akash
Node.js
VS code

① Promise } → (+)ve → resolve ( )
→ (-)ve → reject( )

P.then ( ) .catch ( ) .finally ( )

② (P1, P2, P3 — Pn)
wait } all are resolve → Next step

Promise.all([P1, P2 — Pn]).then( )
.catch( )

③ Promise.any([P1, P2 — Pn]).then( ).catch( )

Async Await

inside

in our practical exercises)

Before function definition

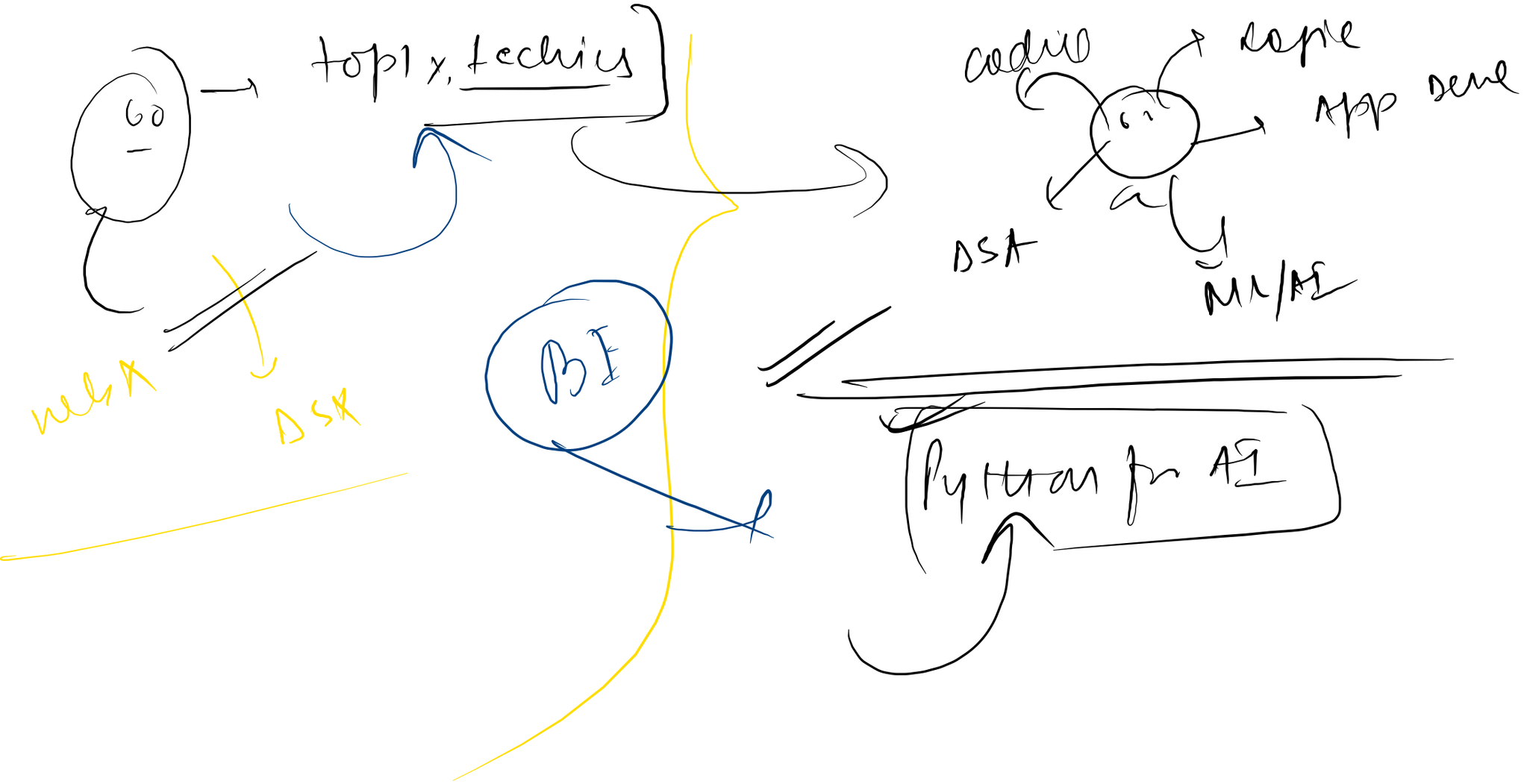
Error Exception Handling

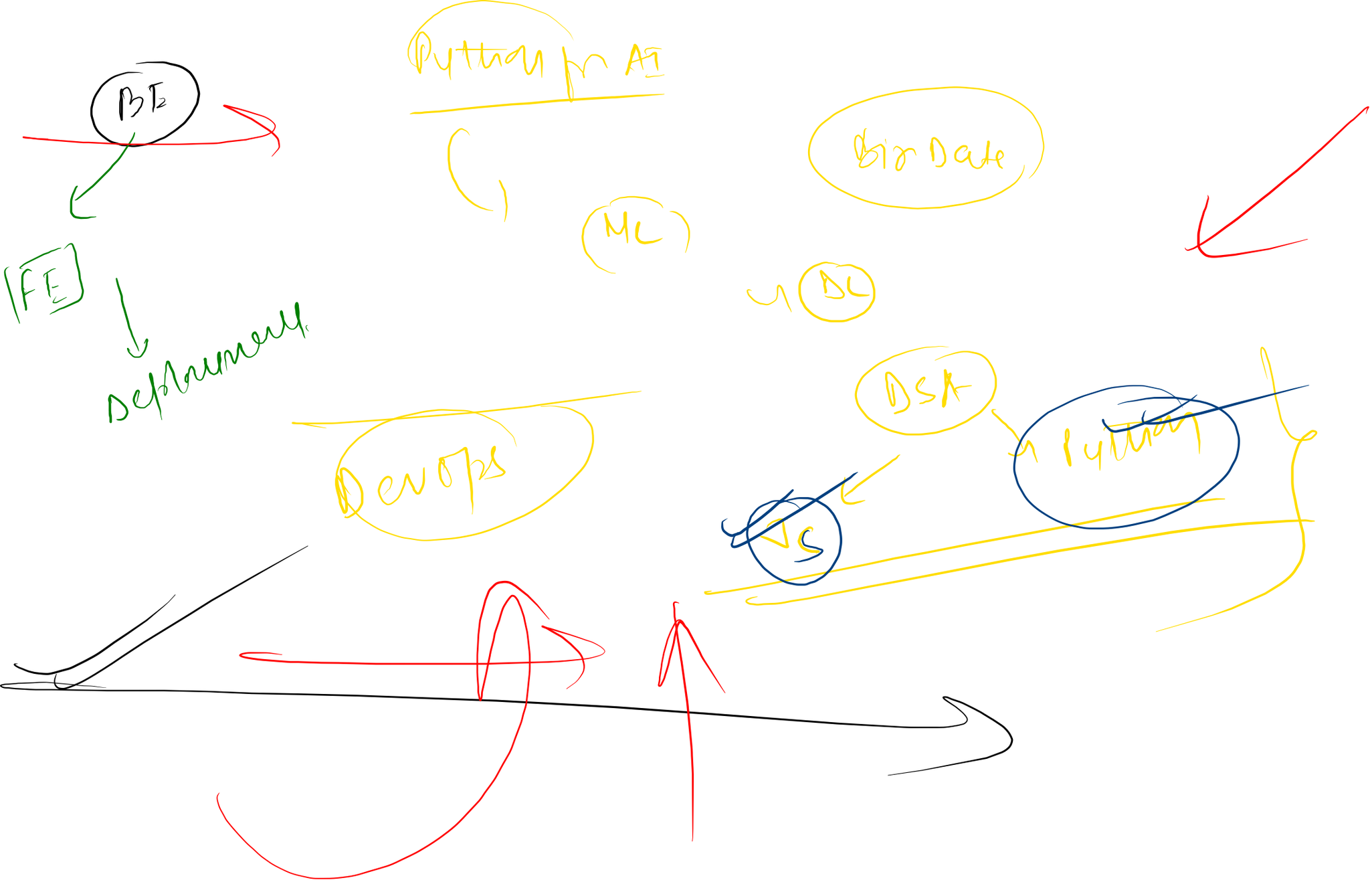
Try {

}

} catch(exn) {

}

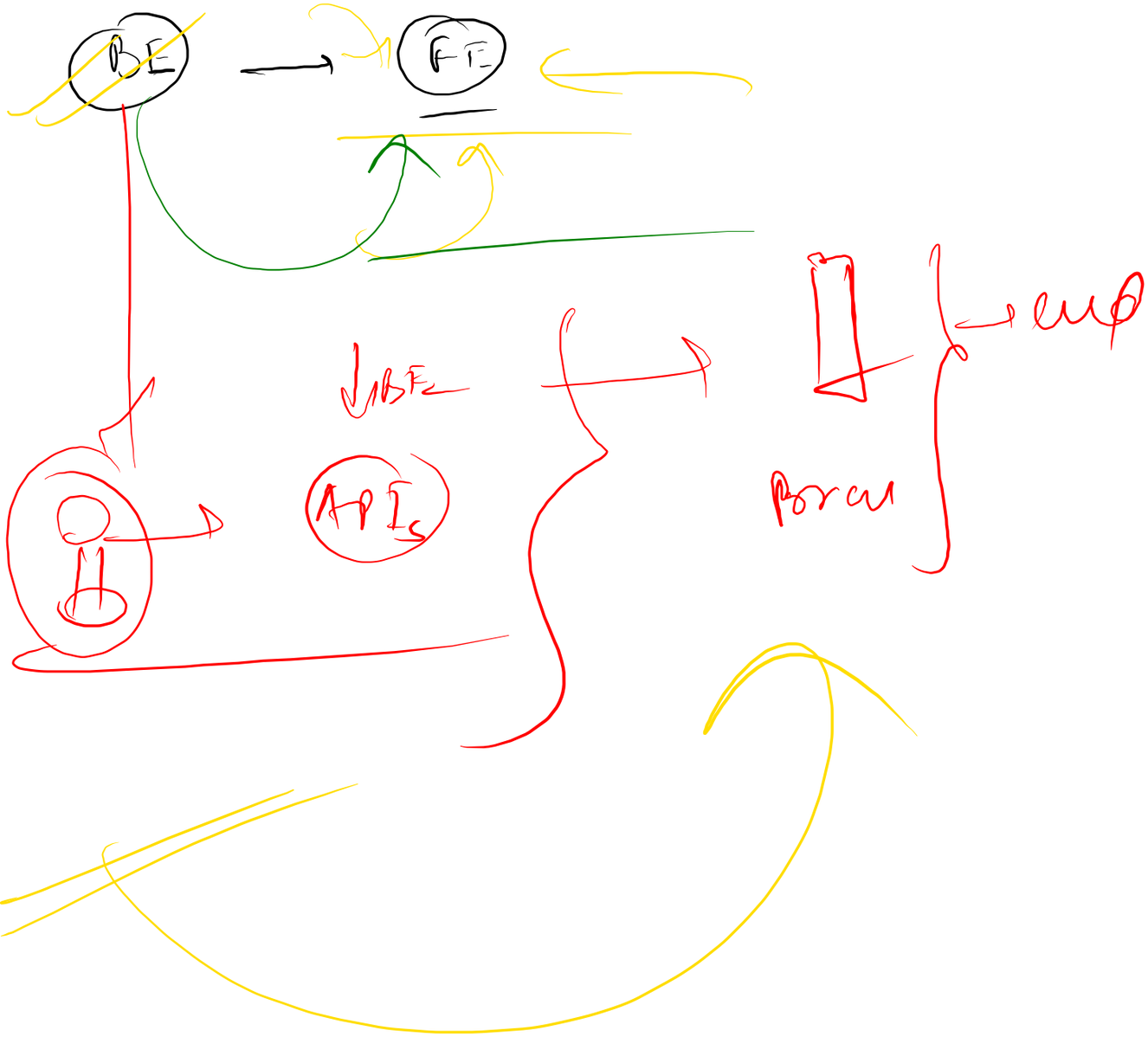

GO
top1 x, techics
web X
DSX
BI
coding
scope
App Dev
DSA
ML/AI
Python for AI


BE
FE
Python for AI
ML
Big Data
DL
DSA
Python
Devops
Vs

DSA

JS

Python

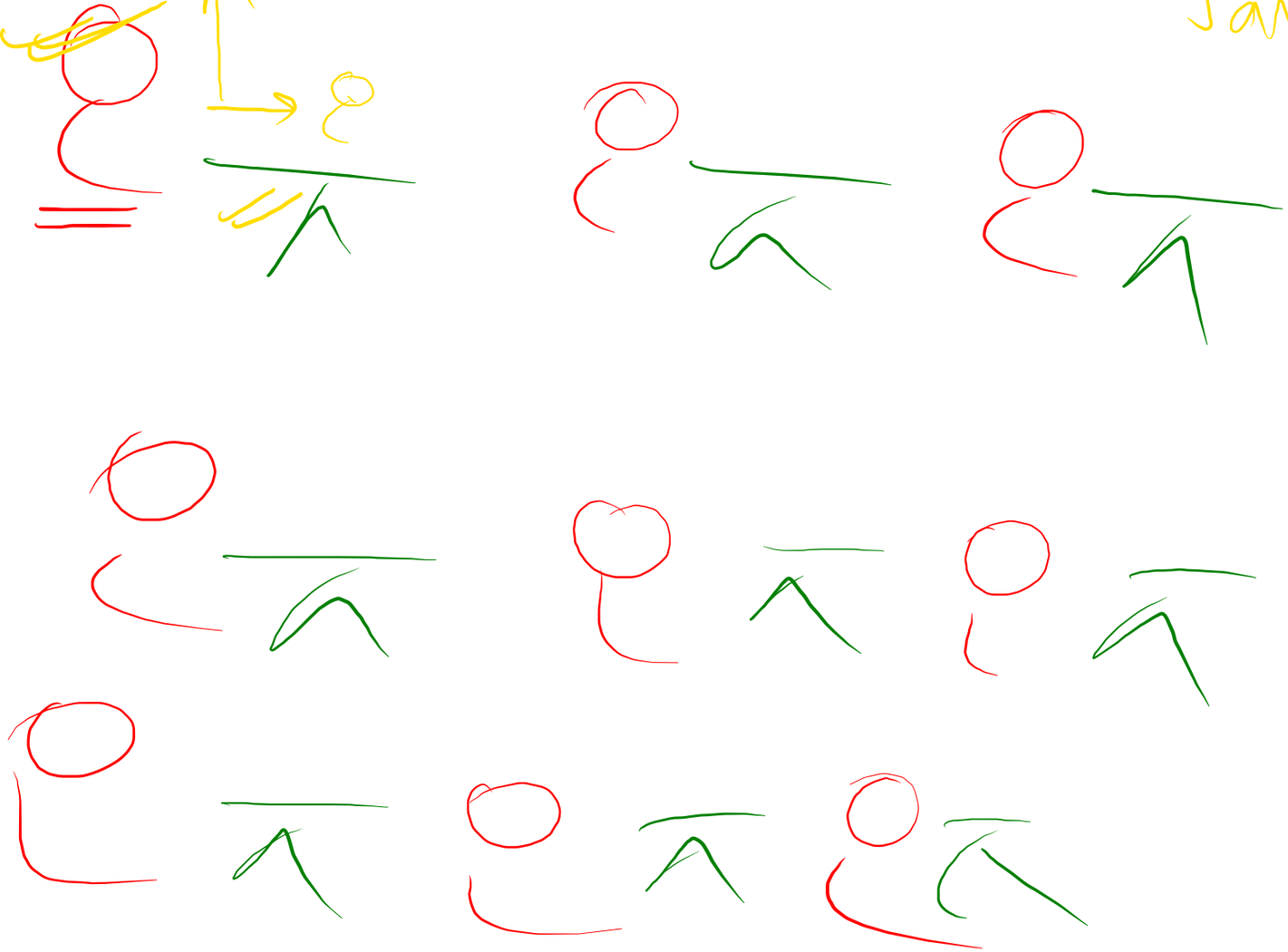

BE
FE
BE
APIs
Porcu
emp

Sym/ Blum way Restaurant ← Ansari

Jana

multicultural

Jana/Ch



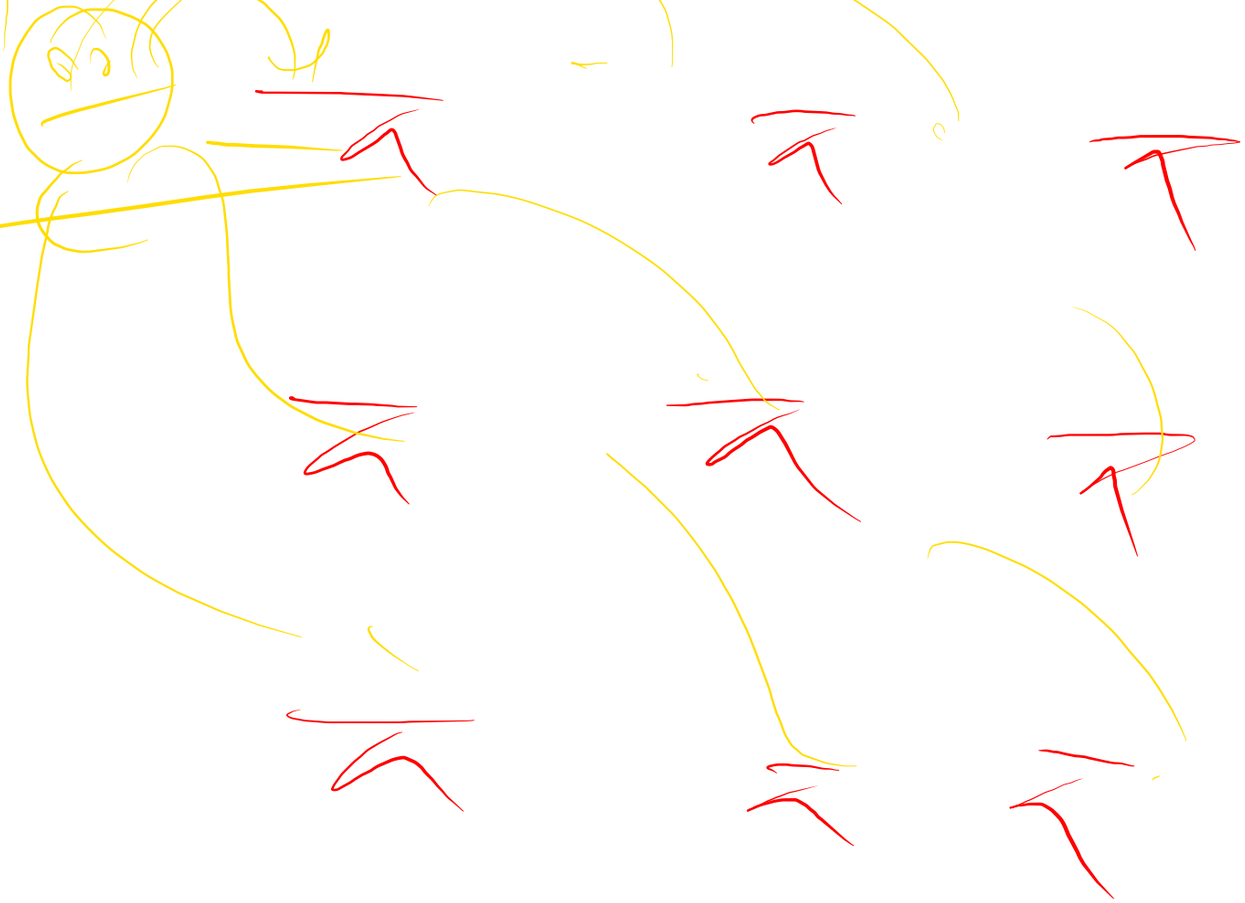
(JS)

Poor Restaurant Owner

Async

Promise

Async/Await



video } every Alternate day

Day 0 → Live }

Day 1 → Idea

4P

Day 2

Live

4P

Day 3

Project

SW Live

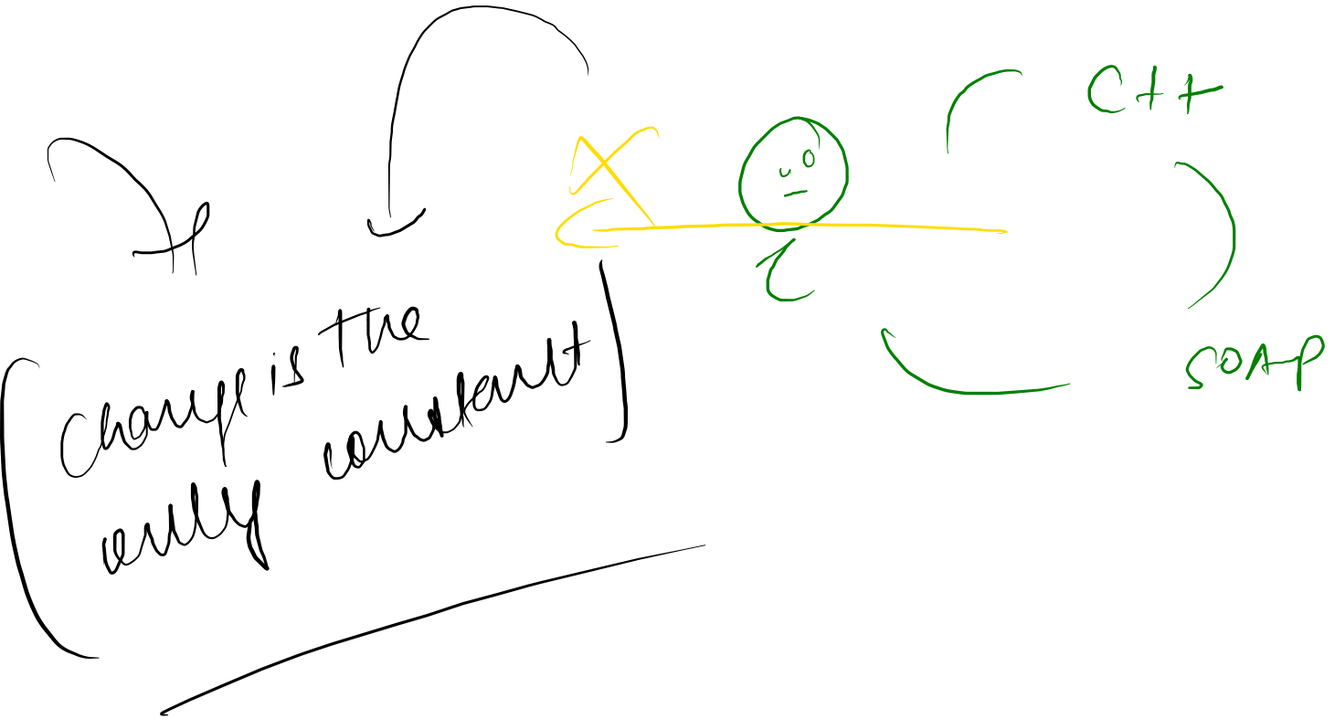
Pwskill.com

course → content

Asy/Await | Promise

same performance

Preferred


C++
SOAP
Change is the only constant

Fundamentals

PL

standard Book

DS

N/W

DB

X

evolving

Python Django

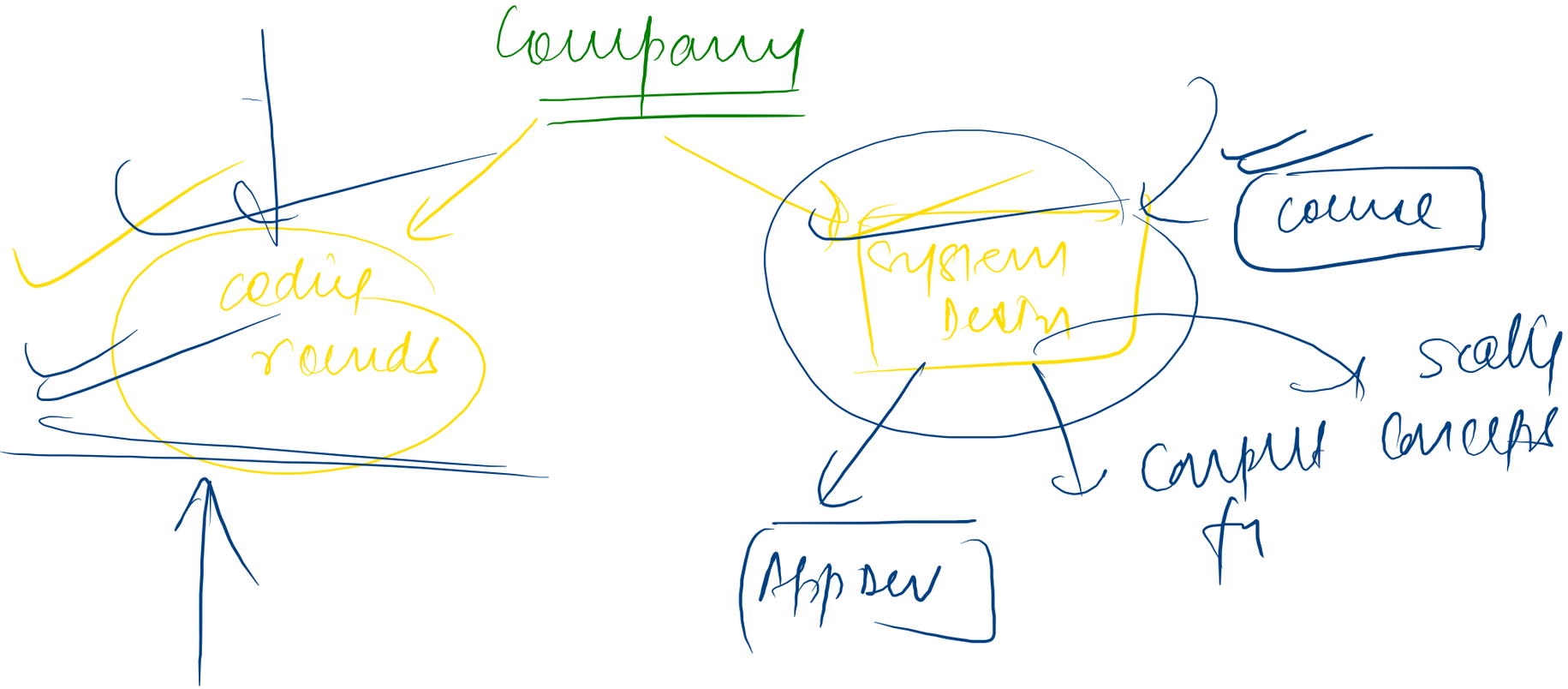
Node

React

Python DSA → taking you toward CP

complete

top 1 percentile no-nursing


Company
coding rounds
System Design
course
scaling
App Dev

if else

sync code

Promise → Asyn code

Typescript

(JS) → (type)

let name = 'Vishal'

typeX

(JS) → typescript → Java

ecommerce backend

BE

BE

FE

- calculation
- Maths
- logic
- Complex

1. Dreams
2. Aviations
3. Repeatitive work

Full stack developer

Google → top logical interview puzzles

Weak → system, prop, DSA, puzzle

top 1%

NeedHelp

Kabir Documents

Books!

Why?

Every respect fark!

Tech

1-2 years

written

Late

Project] → (Async / Await)