

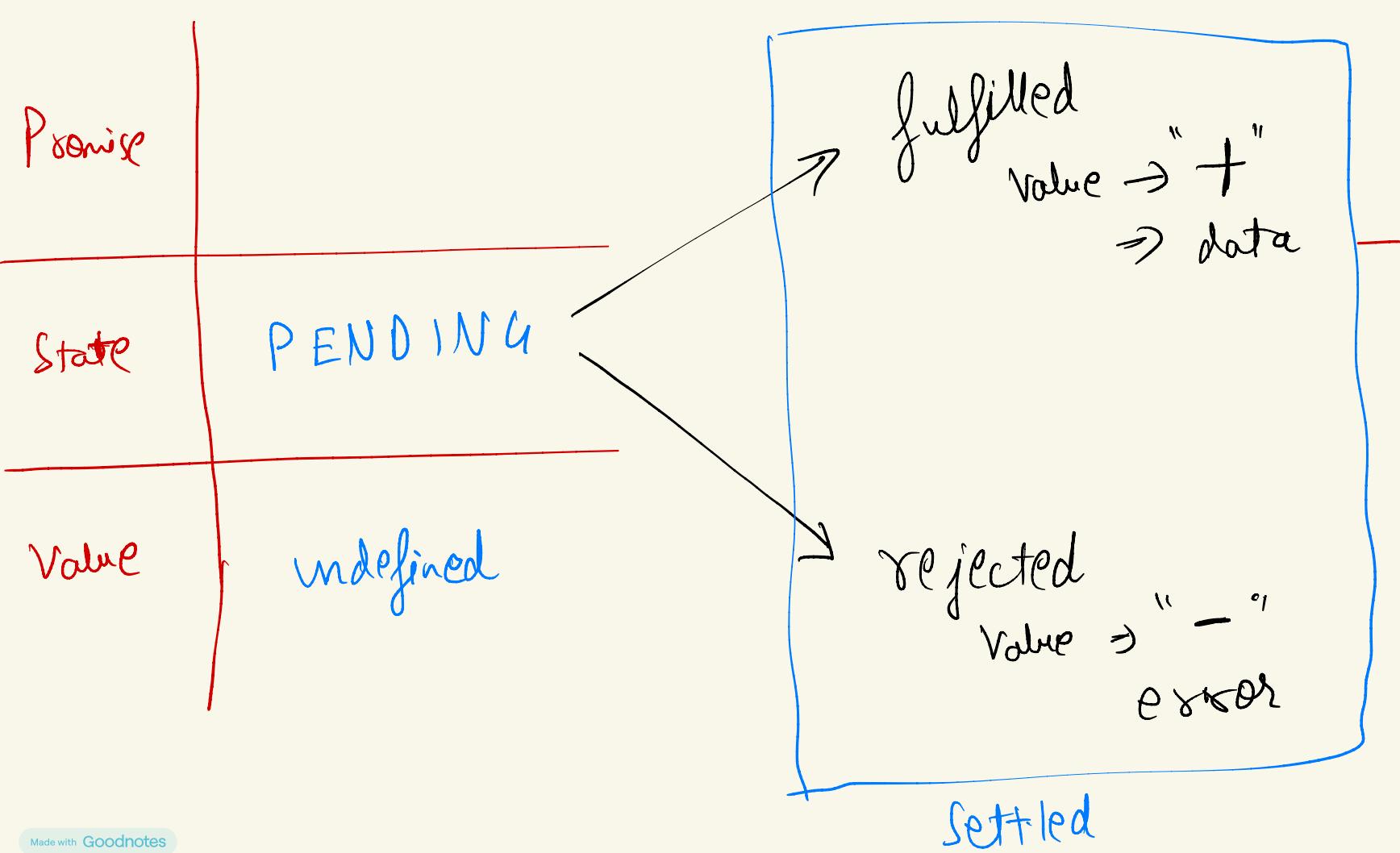
Promise → a promise basically represents upcoming completion or a failure of an asyc event.
& it's resulting value.

```

graph TD
    State["* State"] --- Pending["* Pending  
(S state)"]
    State --- Value["Value"]
    Pending --- failed["failed"]
    failed --- PersonalComputer["- Personal  
Computer"]
    Value --- PlayStation1["PlayStation"]
    Value --- PlayStation2["PlayStation"]

```

The diagram illustrates a state machine for a game board. The root node is "State". It branches into "Pending" (red) and "Value" (purple). "Pending" has a red double slash below it. "Value" branches into two "PlayStation" nodes (purple). The second "PlayStation" node also has a red double slash below it. A horizontal line connects the two "PlayStation" nodes. Below the "Pending" node, there is a red double slash, followed by a red arrow pointing to "failed" (black), which then points to "Personal Computer" (black).



```
const p1 = new Promise((res, rej) => {
    const condition = true;
    if (condition) res("here the promise is resolved");
    else rej("promise failed");
})
```

-
- p1. then((data) => console.log(data))
 - . catch ((err) => console.log(err))

res & rej

Some piece
of code

```
for(let i=0; i<10000; i++) {
```

```
    console.log("Some Sync");
```

}

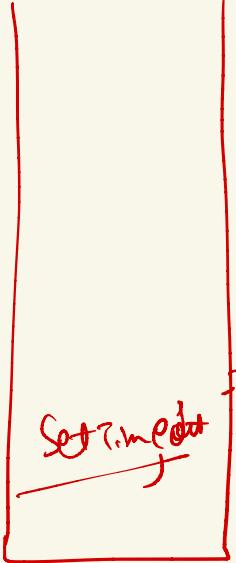
Sync

async

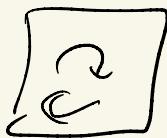
Sync

- ① → "Some Sync Code" 10,000 times
→ resolved value / rejected } → async

-
- ② → resolved value / rejected value
→ "Some Sync Code", 10,000 times } → Sync



Call Stack



event
loop

setTimeout(l) => {
 console.log("SetTimeout");
}, 0);



p1.then((data) => console.log(data));

.catch((err) => console.log(err));

// thread blocking code

() => { };

call back queue

high priority
call back queue
(micro task queue)

```
7 const p2 = new Promise((res, rej) => {
8   console.log("Hi I am inside promise");
9   const condition = true;
10  if (condition) {
11    setTimeout(() => [
12      res("resolved");
13    ], 1000);
14  } else rej("rejected");
15);
16
17 for(let i=0;i<100000;i++){
18   console.log('some sync code');
19 }
20
21 p2.then(data) => console.log(data)).catch(
22
```



1 sec

$\Rightarrow \{ \text{res}(\text{"resolved"}) \}$

Call Stack

" Hi I am inside promise"
" some sync code "
" resolve ",

res("resolve");