Promise.all()

* It takes the all promises in an array and if all the promises will be succeed then only, it proceed.

Promise.allsettled()

* It doesn't care whether the promise is succeed or failure.

suppose

([P_1, P_2, P_2])

P_2 => error

so it will return like

([val1, err, val2])

Promise.race =>

~~Again~~ ~~[illegible]~~

suppose, ([p_1, p_2, p_3])
↓ ↓ ↓
2s 1s 3s
✓

After 1s, it will give the result of second promise, (means give the value of first settled promise)

suppose, after 1s p_2 will give error, so error will ~~come~~ return ~~of~~ for first settled promise.

Promise.any =>

suppose, ([p_1, p_2, p_3])
It is similar to race, but it will wait for the first promise to resolved, but

Date:

in promise.race() doesn't matter & resolved or rejected.

If every promise will fail, return result will be "aggregate error" => error of all the error in array.

[err 1, err 2, err 3]

```
const p1 = new Promise(function(res, rej){
    setTimeout(function(){
        resolve("p1");
    }, 2000);
});
```

```
const p2 = new Promise(function(res, rej){
    setTimeout(Functio n(){
        resolve("p2");
    }, 1000);
});
```

Same for P3 (3000 -> time)

```
Promise.all([P1, P2, P3]).then(function(e){
```

Date:

console.log (res);
})

Output => ['~~Pro~~.p1', 'p2', 'p3']

for Promise.all

Promise.allSettled => (It will give you array of objects)
Output

```
[
 {status: '~~re~~fulfilled', value 'Promise1'}
 {status: '~~fail~~fulfilled', value 'Promise2'}
 like that
]
```

Settled mean It can be failure or resolved!!

If all will give errors then in promise.any(), case aggregate error.

[AggregateError: All promises were rejected] {

Date:

[errors] : ['p1', 'p2', 'p3']

3

What is AggregateError?

Now explain this above eg.