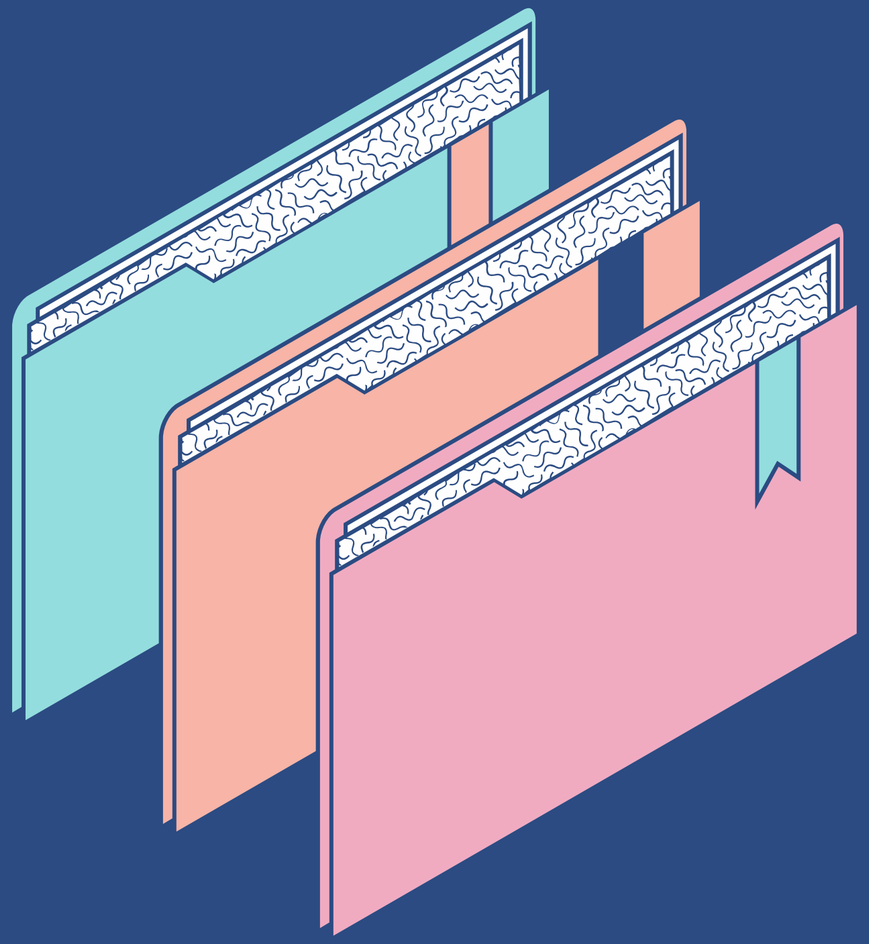
Javascript 3

Putra Prasetya

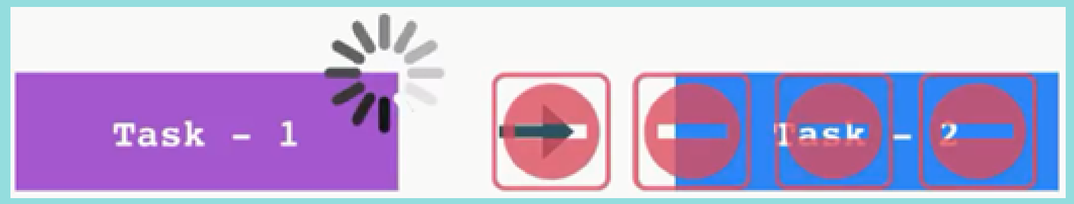


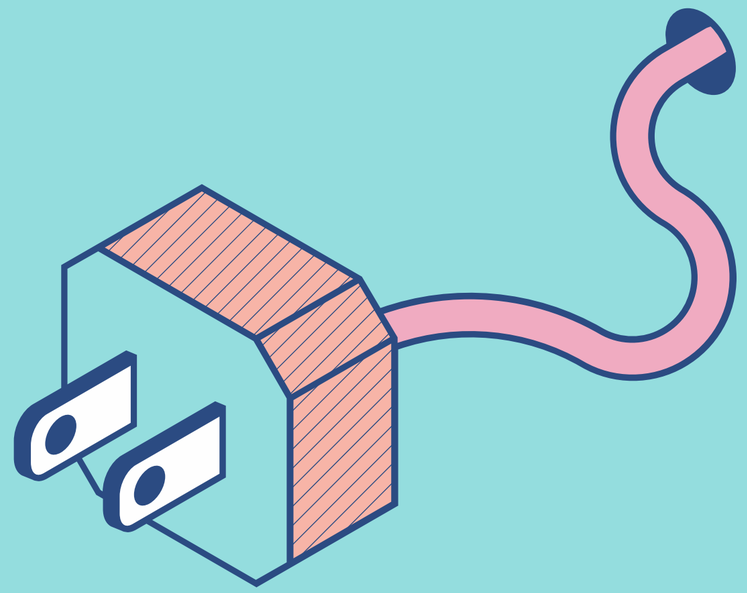
Objective

- Asynchronous
- Callback Asynchronous
- Promise
- Async Await
- Try catch

Synchronous

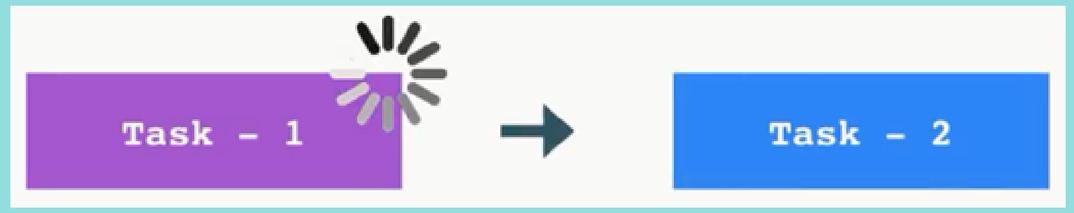
Adalah arsitektur pemblokiran, sehingga eksekusi setiap operasi bergantung pada penyelesaian operasi sebelumnya. Setiap tugas memerlukan jawaban sebelum melanjutkan ke iterasi berikutnya.

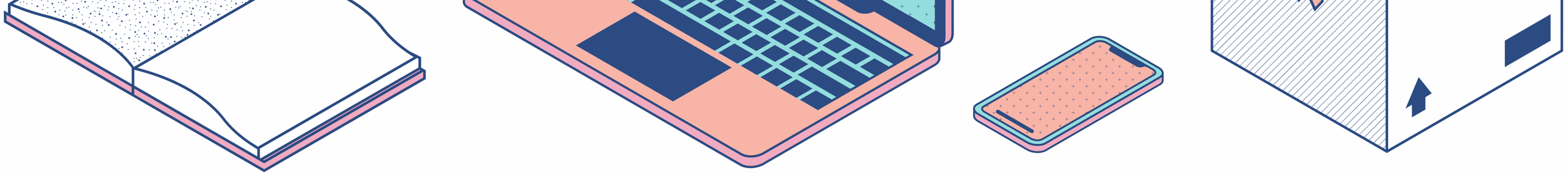




Asynchronous

Adalah arsitektur non-blocking, pelaksanaan tugas tidak bergantung pada tugas lainnya. Tugas dapat berjalan secara bersamaan.





Synchronous vs Asynchronous

- Single-thread (hanya satu operasi atau program yang akan dijalankan pada satu waktu).
- Blocking (mengirimkan satu permintaan ke server dalam satu waktu dan akan menunggu permintaan tersebut dijawab oleh server).
- Lebih lambat dan lebih metodis.

- Multi-thread (operasi atau program dapat berjalan secara paralel).
- Non-blocking (mengirimkan banyak permintaan ke server).
- Meningkatkan *throughput* karena beberapa operasi dapat dijalankan secara bersamaan.

Callback Asynchronous

- Fungsi yang diteruskan ke fungsi lain yang mulai mengeksekusi kode di latar belakang.

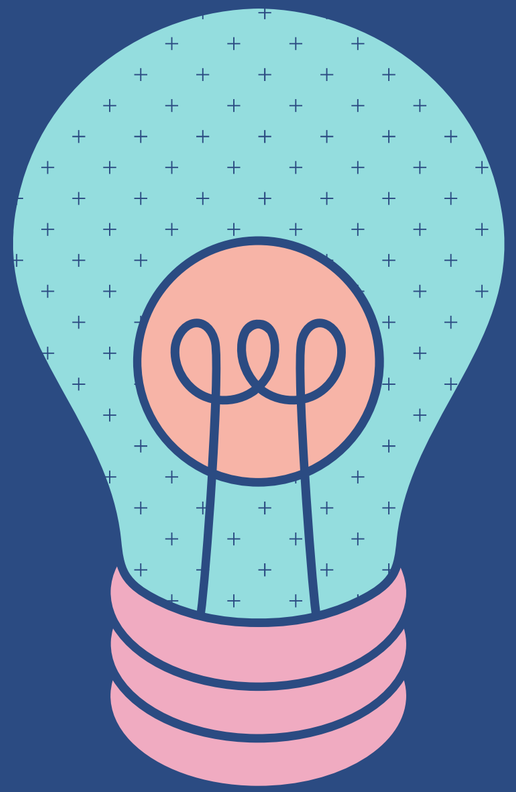
- Biasanya, ketika kode di latar belakang selesai, fungsi callback asynchronous dipanggil sebagai cara untuk memberi tahu dan meneruskan data ke fungsi callback bahwa tugas latar belakang telah selesai.

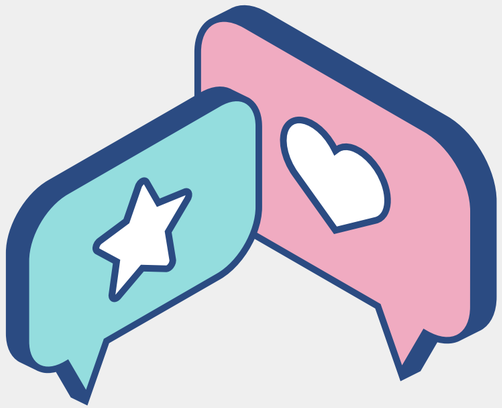
- Contoh :

```
console.log('fired first');
console.log('fired second');

setTimeout(()=>{
    console.log('fired third');
},2000);

console.log('fired last');
```

Output

```
fired first
fired second
fired last
fired third
```





Promise

- Objek Promise mewakili penyelesaian (atau kegagalan) operasi asynchronous dan nilai yang dihasilkannya.
- Promise adalah perantara untuk nilai yang belum tentu diketahui saat promise dibuat.
- Hal ini memungkinkan metode asynchronous mengembalikan nilai seperti metode synchronous: alih-alih langsung mengembalikan nilai akhir, metode asynchronous mengembalikan promise untuk memberikan nilai di masa mendatang.

State Promise

Layaknya membuat janji, terdapat tiga kemungkinan keadaan (state) yang dapat terjadi di dalam janji (promise) :

1

Pending

Keadaan awal, tidak terpenuhi atau ditolak.

2

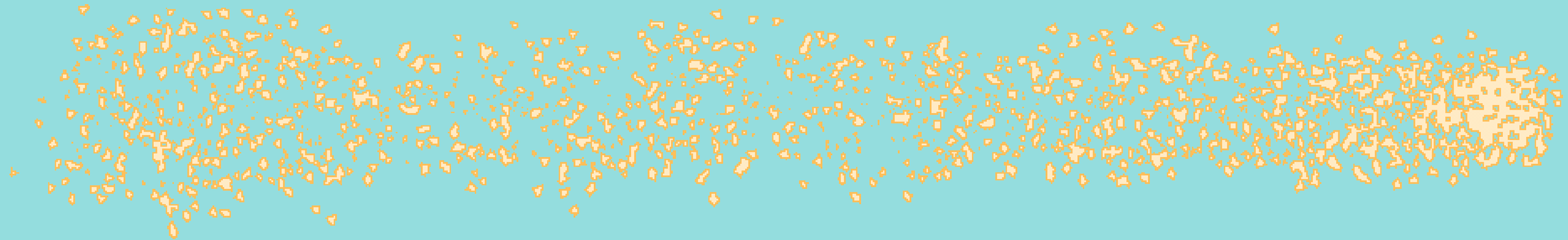
Fulfilled

Operasi berhasil diselesaikan

3

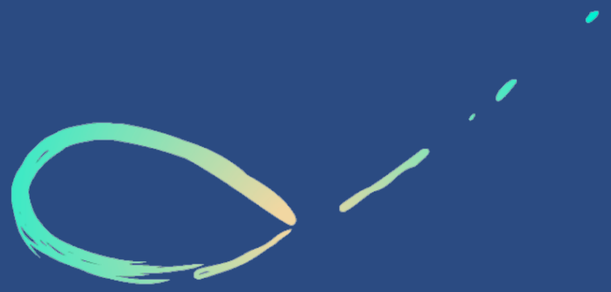
Rejected

Operasi gagal



Promise

KODE ⟶



```
const axios = require('axios');

let nilai = 'https://jsonplaceholder.typicode.com/users'
const janjian = () => {
 return new Promise((resolve, reject) => {
  axios.get(nilai)
   .then((response) => {
     resolve(response.data[0].name)
   })
   .catch((err) => {
     reject(err)
   })
 })
}

janjian(nilai)
 .then((result) => {
  console.log(result);
 })
 .catch((err) => {
  console.log('hasil gagal');
 })

// Output : Leanne Graham
```

Async / Await

Async -> mengubah function menjadi asynchronous.

Await -> menunda eksekusi hingga proses asynchronous selesai.

```
const axios = require('axios');

async function getData() {
  try {
    let nilai =
'https://jsonplaceholder.typicode.com/users'
    const data = await axios.get(nilai)
    console.log(data.data[0].name);
  } catch (e) {
    console.log('data error');
  } finally {
    console.log('Ini hasil nya');
  }
}

getData();

// Output
Leanne Graham
Ini hasil nya
```

Try Catch Finally

Try -> Blok kode yang akan dijalankan (untuk dicoba).

Catch -> Blok kode untuk menangani kesalahan apapun.

Finally -> Blok kode untuk dijalankan terlepas dari hasilnya.

```
const axios = require('axios');

async function getData() {
  try {
    let nilai =
'https://jsonplaceholder.typicode.com/users'
    const data = await axios.get(nilai)
    console.log(data.data[0].name);
  } catch (e) {
    console.log('data error');
  } finally {
    console.log('Ini hasil nya');
  }
}

getData();

// Output
Leanne Graham
Ini hasil nya
```

Terimakasih