

# Javascript



## Week 01 – Day 5

author : kang dian  
Principal Trainer



# Roadmap NodeJS Full Stack

Week I

- Data Type
- JS Memory
- Variable Scope
- String Manipulation
- Number
- Operator
- Functions
- Conditions
- Iteration
- Data Structure
- Functional
- Object Type
- Async Await
- Coding Challenge

Week II

- Database
- NodeJS Fundamental
- RestFul API
- Raw Query
- Unit Testing
- Debugging
- Create-React-App
- HR Project I

Week III

- Build Project with Webpack
- HR Project II
- Server Side Rendering
- TailwindCSS
- Assignment Mini Project

Week IV

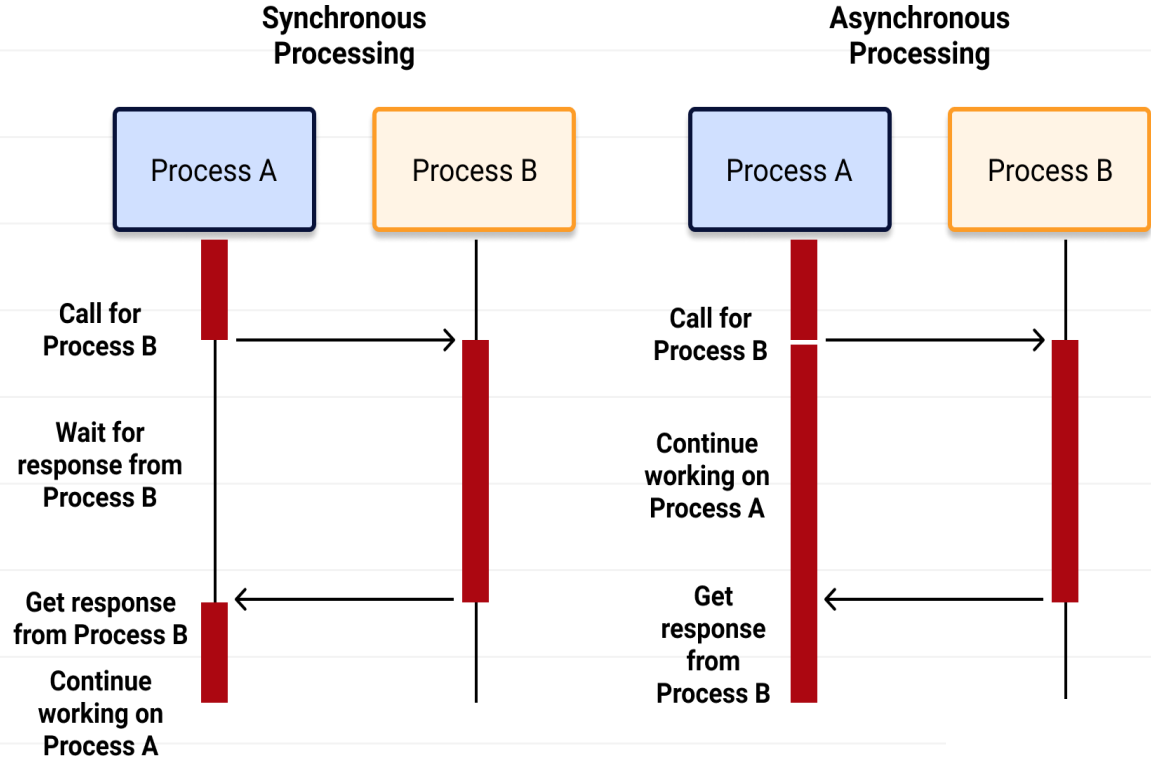
- Redux
- GraphQL

Week V-VII

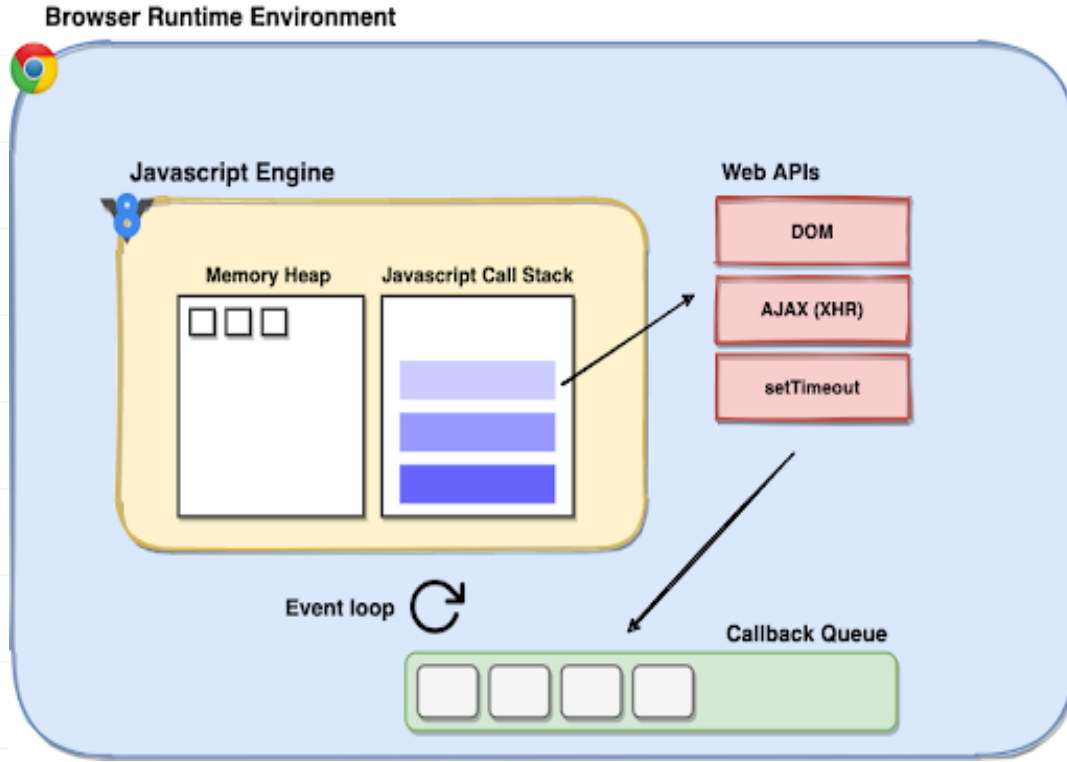
- Mini Projects

# Syncronous VS Asynchronous

- Sync & Asynch adalah programming model untuk melakukan proses komputansi dengan cara yang berbeda.
- Sync akan meng-eksekusi sebuah task (process) dalam satu waktu sampai selesai. Contoh process A akan meng-eksekusi process b, process a akan mengunggu pekerjaan process b sampai selesai.
- Berbeda dengan sync, async meng-eksekusi sebuah process secara parallel tanpa menunggu process b selesai.
- Kelebihan dari async adalah dapat menghandle multi task, sehingga performansi dalam menyelesaikan multi task bisa lebih cepat.
- Task yang penyelesaiannya tidak menentukan kapan selesainya, maka harus diperlakukan menjadi task async.



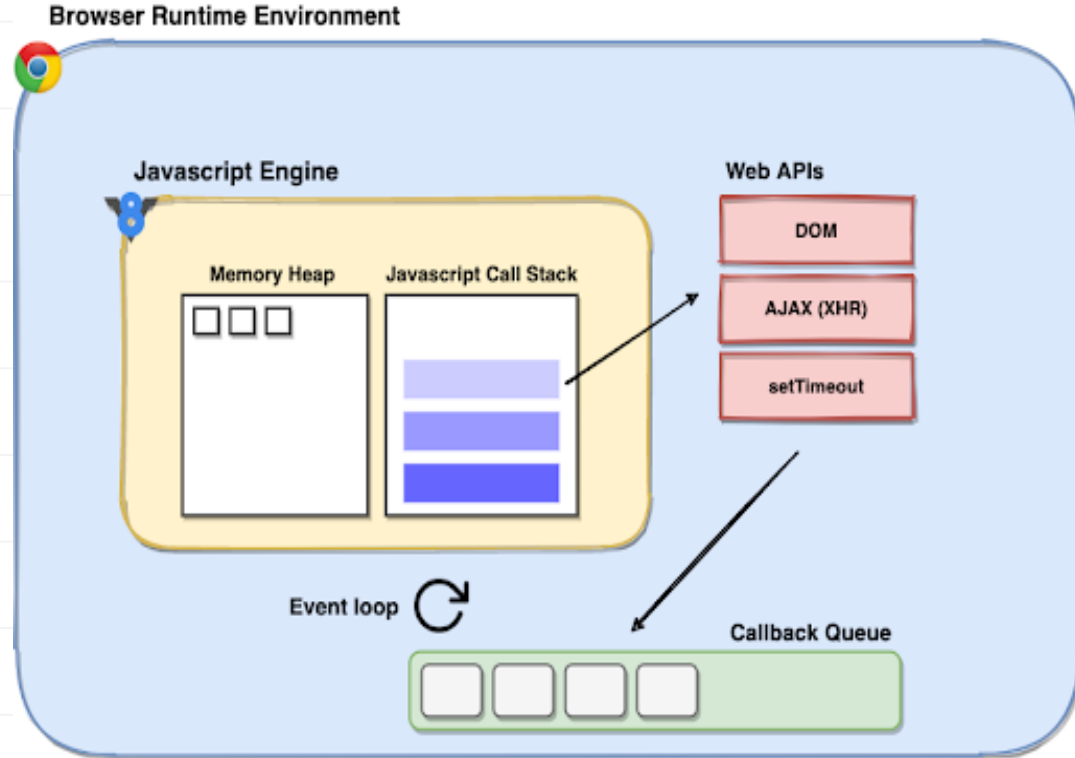
# Javascript Environment Main Part



- Call Stack adalah part memory untuk menyimpan variable primitive data type dan variable yang menyimpan address value ke heap memory.
- Memory Heap adalah part memory untuk menyimpan non primitive data type.
- Web APIs bagian dari browser untuk menyimpan asynchronous operation seperti DOM, AJAX, setTimeout, Geolocation, LocalStorage
- Callback Queue adalah antrian untuk menyimpan task function asynchronous yang menunggu panggilan Event Loop untuk dilempar ke Call Stack
- Event Loop adalah process di js runtime yang akan men-check Call Stack, jika Call Stack dalam keadaan kosong, maka Event Loop akan panggil task antrian berikutnya

# How JS Runtime Enviroment Work?

- Javascript memiliki kemampuan untuk execute script code secara synchronush dan asynchronush
- Task function synchronush akan langsung di tempatkan di Call Stack untuk segera di execute.
- Task function asynchronush akan ditempatkan di Callback Queue, jika Event Loop mendeteksi Call Stack dalam keadaan kosong (empty), maka function task di Callback Queue akan dilempar (push) ke Call Stack untuk di eksekusi
- Task functions asynch hanya akan di lempar ke Call Stack jika Call Stack dalam keadaan kosong.
- Call Stack menggunakan metode LIFO (Last In First Out)



Call Stack & Heap adalah memory untuk menyimpan variable primitive data type dan non primitype data type

# Script Code Sync vs Async

// synchronouse script code

```
const antrian1 =()=>{  
  console.log("Antrian 1 : Beli KFC paket A");  
}  
  
const antrian2 =()=>{  
  console.log("Antrian 2 : Beli KFC paket B");  
}  
  
const antrian3 =()=>{  
  console.log("Antrian 3 : Beli KFC paket C");  
}
```

// call function

```
antrian1();  
antrian2();  
antrian3();
```

## Output

```
Antrian 1 : Beli KFC paket A  
Antrian 2 : Beli KFC paket B  
Antrian 3 : Beli KFC paket C
```

//script code asynch

```
const antrian1 = () => {  
  console.log("Antrian 1 : Beli KFC paket A");  
}  
  
const antrian2 = () => {  
  setTimeout(() => {  
    console.log("Antrian 2 : Beli Sup Ayam");  
  }, 2000);  
}
```

```
const antrian3 = () => {  
  console.log("Antrian 3 : Beli KFC paket C");  
}
```

```
const antrian4 = () => {  
  setTimeout(() => {  
    console.log("Antrian 4 : Beli Burger");  
  }, 1000);  
}
```

//call functions

```
antrian1();  
antrian2();  
antrian3();  
antrian4();
```

Jika function antrian4() panggil setTimeout(), maka js akan menyimpannya di antrian (callback queue)

## Output

```
Antrian 1 : Beli KFC paket A  
Antrian 3 : Beli KFC paket C  
Antrian 4 : Beli Burger  
Antrian 2 : Beli Sup Ayam
```

# Evolution JS Handle Async



Callback

- Callback sudah dipakai diawal javascript lahir

Promise

- Mulai dikenalkan di ES6 (2015)

Asyn Await

- Mulai dikenalkan di ES8 (2017)

## Alasan kenapa sebuah Task harus diperlakukan Async :

- Jika sebuah task tidak tentu kapan selesai proses nya, maka jadikan task tersebut menjadi async. Hal ini karena pertimbangan network, bandwith, proses I/O (input/output)
- Example : Ajax, DOM, Geolocation, Localstorage, Event, Request API ke server. Operation diatas adalah contoh yang di execute secara async

# Callback (Before ES6)

- JS akan memperlakukan sebuah task menjadi task async jika task tersebut memiliki parameter yang isinya berupa function (callback).
- Kelemahan metode ini adalah, jika ada ketergantungan antara task maka akan sulit mendeteksi error nya dan sulit dibaca, yang biasanya dikenal dengan istilah Callback Hell.



```
// callback script code
const callCenter147 = () => {
  pesan("Selamat Datang Di Layanan Telkom 147", ()=>{
    pesan("Pilih angka 1 untuk indonesia", ()=>{
      pesan("Tekan angka 1 untuk pendaftaran", ()=>{
        pesan("Tekan angka 2 untuk keluhan", ()=>{
          pesan("Tekan angka 3 untuk membatalkan", ()=>{
            })
          })
        })
      })
    })
  });
}

const pesan = (message, callback)=>{
  setTimeout(() => {
    console.log(message);
    callback();
  }, 2000);
}

// call funtion
callCenter147();
```

Function pesan di parameter kedua memanggil data type function. JS secara otomatis akan memperlakukan sebagai task function async.




# Promise


- Dikenalkan mulai ES6(2015). Lebih mudah dibaca dibanding callback.
- Object Promise memiliki 2 parameter callback yaitu resolve dan reject, resolve jika success, reject jika error.

```
// contoh promise
const mypromise = new Promise((resolve, reject) => {
  if ((Math.random() * 10) <= 5) {
    resolve('Lunas yaa...!');
  }
  reject(new Error('Belum ada uang, ntar sok yah.. '));
})
```

```
// call function mypromise
mypromise
  .then(response=>console.log(response))
  .catch(err=>console.log(err));
```



Keyword .then digunakan untuk menghandle return response success dari callback resolve



Keyword .catch adalah function untuk menghandle error yang dikirim dari callback reject

# Async With Promise

- Dikenalkan mulai ES8(2017).
- Lebih mudah dibaca dan mengurangi kompleksitas callback yang digunakan oleh Promise.
- Keyword Async & Await digunakan terpisah di script code.
- Async dapat juga memanggil promise.

```
// async with promise
async function getNomorAntri(nomor){
  if(nomor <= 10){
    return Promise.resolve(10);
  }else{
    return Promise.reject(new Error("Antrian habis"))
  }
}

const order = () => {
  console.log("Order KFC paket A");
}

//call function
getNomorAntri(12)
  .then(res => console.log(res))
  .catch(error => console.log(error.toString()));

order();
```

# Async With Await

```
const getNomorAntri=(nomor)=>{
  return nomor;
}

const pilihPaket = (nomor,paket)=>{
  if (nomor <= 0 && isNaN(nomor)){
    return "silahkan antri"
  }
  if (paket === "A"){
    return "KFC Paket A"
  }else{
    return "KFC Paket B"
  }
}
```

```
const tagihan =(paket)=>{
  if (paket === "A"){
    return 25000;
  }else{
    return 24000;
  }
}
```

```
const orderKFC = async(paket)=>{
  const nomorAntri = await getNomorAntri(10);
  const menu = await pilihPaket(nomorAntri,"A");
  const total = await tagihan(menu);
  return [nomorAntri,menu,total];
}

// call order kfc
orderKFC("A").then(res => console.log(res));
```

# Async Await With Promise Chain

```
const getNomorAntri=(nomor)=>{  
  return nomor;  
}
```

```
const pilihPaket = (nomor,paket)=>{  
  if (nomor <= 0 && isNaN(nomor)){  
    return "silahkan antri"  
  }  
  if (paket === "A"){  
    return "KFC Paket A"  
  }else{  
    return "KFC Paket B"  
  }  
}
```

```
const tagihan =(paket)=>{  
  if (paket === "A"){  
    return 25000;  
  }else{  
    return 24000;  
  }  
}
```

```
const orderKFC = async(paket)=>{  
  const [nomorAntri,menu,total] = await Promise.all(  
    [  
      getNomorAntri(10),  
      pilihPaket(10,"A"),  
      tagihan("A")  
    ]  
  );  
  return [nomorAntri,menu,total];  
}  
orderKFC2("B").then(res => console.log(res))
```

# THANKS!

## Any questions?

CREDITS: This presentation template was created by [Slidesgo](#), including icons by [Flaticon](#), and infographics & images by [Freepik](#).

