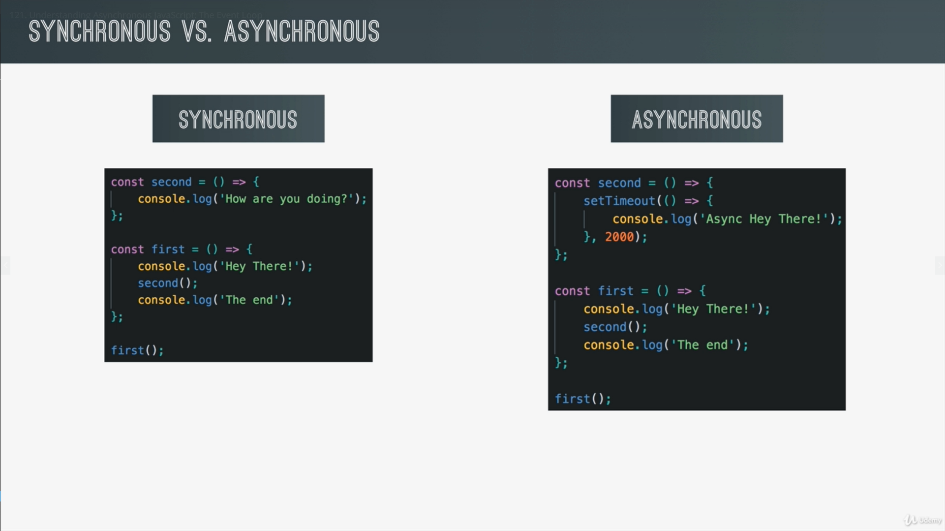
Memahami Asynchronous Javascript

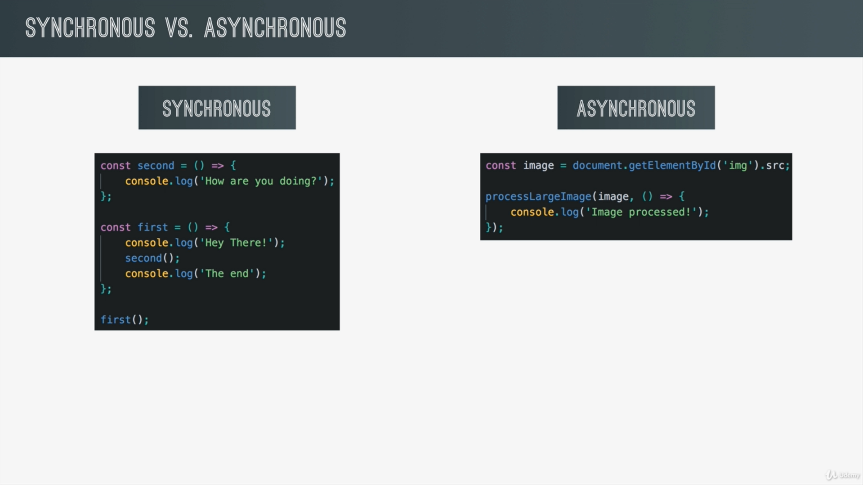
# Contoh sebelumnya



🡪 Code tidak berhenti selama 2 detik.

🡪 Melainkan prosesnya dipindahkan dan baru akan berjalan di console setelah 2 detik.

# Sedikit Penggambaran

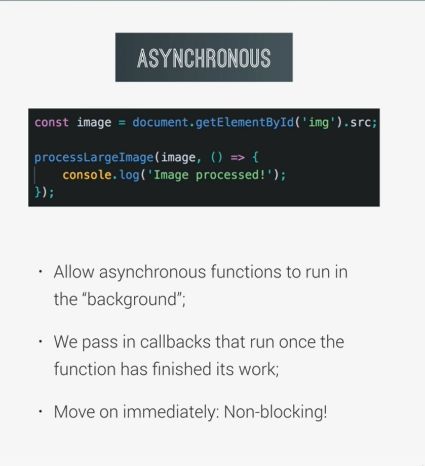


🡪 Coba bayangkan kamu punya proses untuk load image dari satu sumber tertentu.

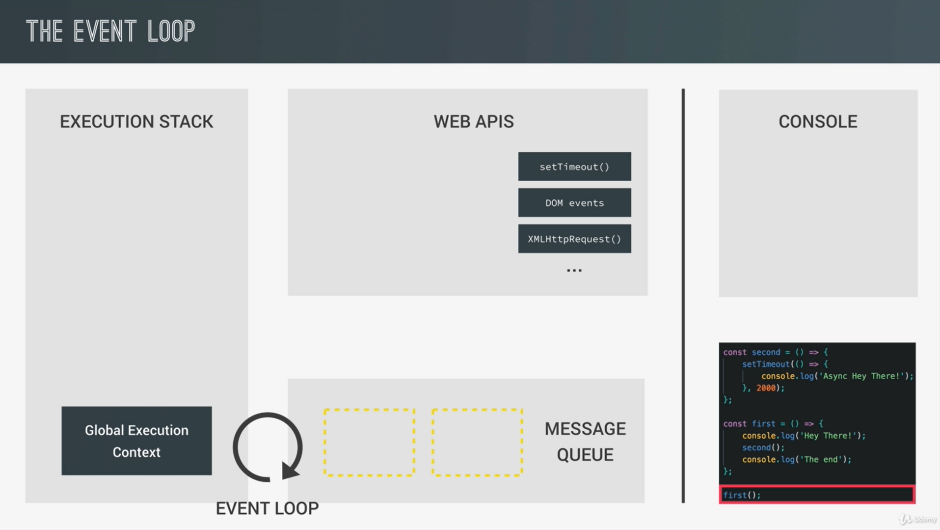
🡪 Tapi akan lebih baik jika user tidak perlu menunggu proses gambar itu selesai di load.

🡪 Jadi yang kita lakukan adalah kita potong prosesnya agar berjalan di satu tempat lain, sehingga di tampilan utama semua proses lainnya dapat berjalan tanpa harus menunggu proses load image tersebut

🡪 Seperti itulah asynchronous code.

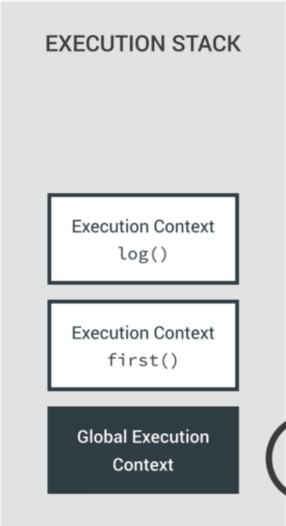
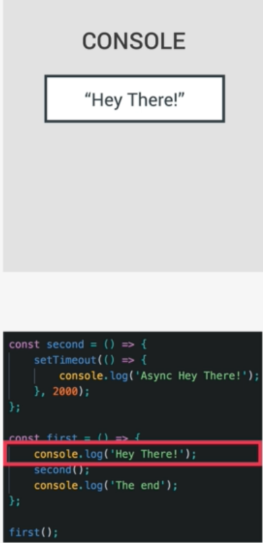


# Dalam Gambaran besarnya



🡪 Fungsi pertama yang dijalankan diawal akan terpanggil ke dalam execution stack.

🡪 Lalu kemudian proses console.log akan dijalankan dan menghasilkan output pada halaman console.

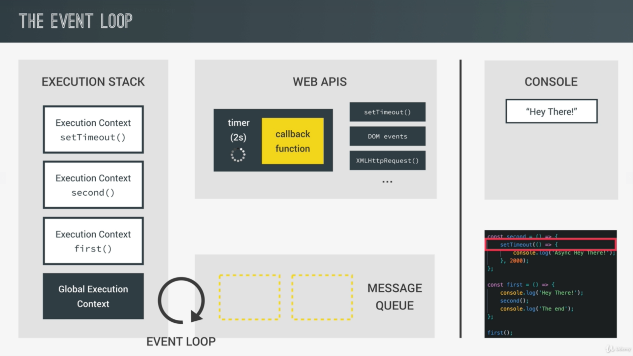
🡪 Dan Execution context log() akan keluar dari proses.

🡪 Kemudian second() function akan masuk ke execution stack.

🡪 Dan setTimeOut akan berjalan

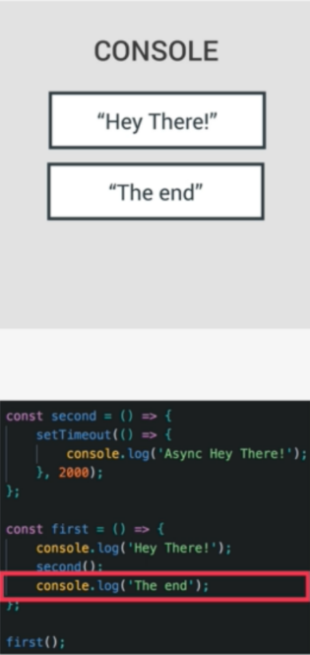
NOTE: SetTimeOut, DOM, XMLHttpReq, GeoLocation, Local Storage dan masih banyak lagi itu merupakan bagian yang berbeda dan dijalankan terpisah dari engine Javascript. Dan akan dijalankan di belakangnya. Kita hanya punya akses kedalamnya karena mereka sama - sama bundle dari Javascript runtime process. Dan inilah alasannya kenapa code tidak terblokir ketika kita menjalankan setTimeout.

🡪 Setelah itu, SetTimeout akan ditempatkan ke Web API, dan kita akan tunggu hingga waktu yang di set telah selesai.



🡪 Dan setelahnya setTimeOut akan dilempar dari execution stack.

🡪 Kemudian console.log() di line berikutnya akan diarahkan dan diproses di execution stack



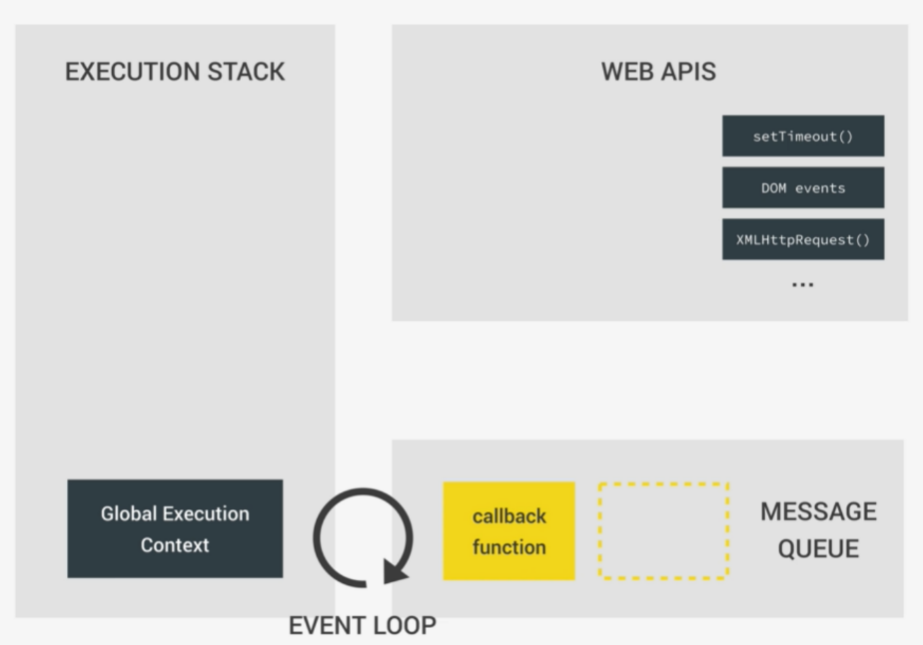
🡪 Kemudian log() dilempar setelah selesai.

🡪 Setelah 2 detik berlalu, kini web api sudah kosong.

🡪 Sama seperti DOM method, callback akan siap untuk dijalankan.

🡪 Bagaimana callback function berjalan ?

🡪 Disitulah tugas dari Event Loop. Ketika diliat di Execution Stack kosong, dia akan mendorong fungsi tersebut ke dalam Execution Stack ketika Message Queue tidak kosong.



🡪 Setelah itu baru dijalankan console.log() dan di print kemudian

