JavaScript Async

Eko Kurniawan Khannedy

License

- Dokumen ini boleh Anda gunakan atau ubah untuk keperluan non komersial
- Tapi Anda wajib mencantumkan sumber dan pemilik dokumen ini
- Untuk keperluan komersial, silahkan hubungi pemilik dokumen ini

Eko Kurniawan Khannedy

- Technical architect at one of the biggest ecommerce company in Indonesia
- 10+ years experiences
- youtube.com/c/ProgrammerZamanNow

Sebelum Belajar

Sudah Menguasai

- HTML
- JavaScript Dasar

Perangkat Lunak

- Google Chrome
  - Allow CORS Plugin : https://chrome.google.com/webstore/detail/allow-cors-access-control/lhobafahddgcelffkeicbaginigeejlf
- NodeJS
  - live-server : https://www.npmjs.com/package/live-server
- Code Editor
  - JetBrains WebStorm
  - VisualStudio Code

Apa yang akan dipelajari?

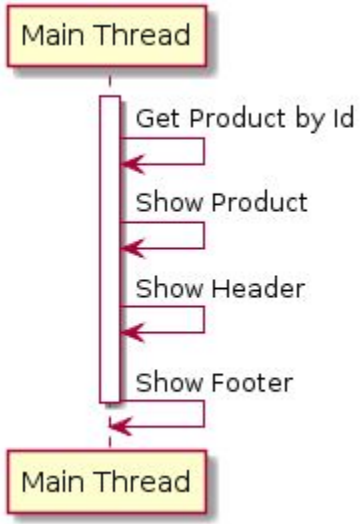
- Asynchronous
- Callback
- AJAX
- Promise
- Async Await
- Web Worker

Apa itu Asynchronous?

Apa itu Synchronous?

- Program dalam JavaScript secara default akan dieksekusi baris per baris
- Secara default, proses di JavaScript akan dieksekusi secara Synchronous, artinya baris selanjutnya akan dieksekusi setelah baris sebelumnya selesai dikerjakan
- Proses Synchronous juga biasa disebut Blocking, karena harus menunggu tiap proses selesai, baru proses selanjutnya bisa dilakukan

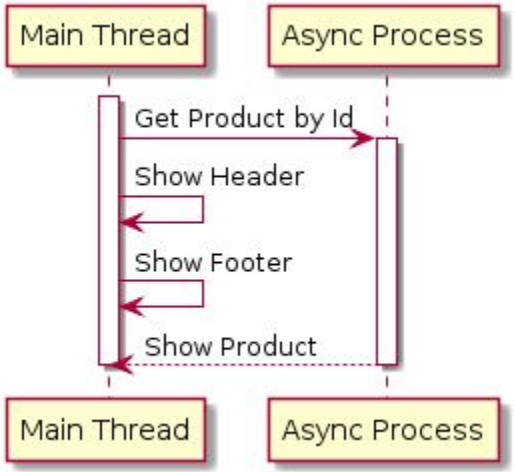
Contoh Synchronous



Apa itu Asynchronous?

- Walaupun secara default proses di JavaScript dieksekusi secara Synchronous, namun kita bisa membuatnya menjadi Asynchronous.
- Berbeda dengan proses Synchronous, pada proses Asynchronous, JavaScript tidak akan menunggu proses tersebut selesai, melainkan JavaScript akan melanjutkan baris selanjutnya, tanpa harus menunggu proses Asynchronous selesai.
- Proses Asynchronous juga biasa disebut Non-Blocking.

Contoh Asynchronous



Callback

Apa itu Callback?

- Callback, merupakan mekanisme untuk memanggil kembali kode yang ada di program dari proses Async
- Callback biasanya dibuat dalam bentuk function, dan function tersebut akan dieksekusi saat proses Async selesai
- Dengan menggunakan Callback, program bisa menerima informasi yang dibutuhkan dari proses yang berjalan secara Async

Contoh Async Method

Ada banyak method Async yang terdapat di JavaScript, yang akan kita bahas kali ini adalah :

- setTimeout(handler, time), digunakan untuk menjalankan proses Async sekali dalam waktu tertentu.
- setInterval(handler, time), digunakan untuk menjalankan proses Async secara periodik dalam waktu tertentu.

setTimeout

```
setTimeout(function () {
     // do something here
}, 5000);
```

setInterval

```
setInterval(function () {
    // do something here
}, 5000);
```

Let's Code

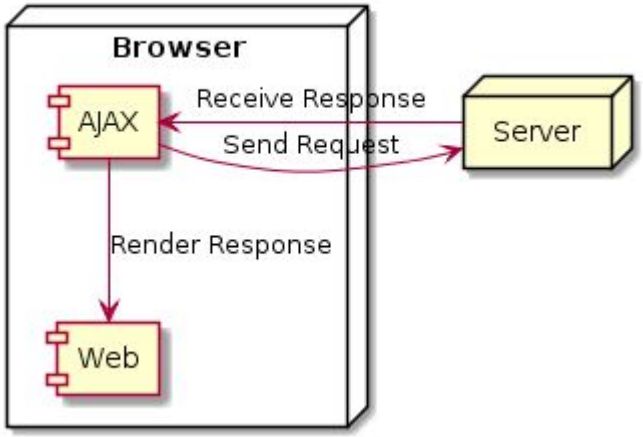
https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/set-timeout.html

AJAX

Apa itu AJAX?

- AJAX singkatan dari **Asynchronous JavaScript and XML**
- AJAX dapat digunakan untuk mengambil data dari server setelah halaman web tampil
- AJAX dapat digunakan untuk mengubah tampilan web tanpa harus me-load ulang web
- AJAX dapat mengirim data ke server secara async di background

Cara Kerja AJAX



Membuat AJAX

```
const ajax = new XMLHttpRequest();

ajax.open("METHOD", "url");

ajax.send()
```

Let's Code

https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/ajax.html

AJAX Callback

AJAX Callback

- AJAX biasanya digunakan untuk mengirim data ke Server atau menerima data dari Server
- Tiap request AJAX yang dilakukan, biasanya kita ingin mendapat informasi response yang diberikan oleh Server
- Kita tidak bisa langsung mengambil response AJAX, karena proses AJAX adalah Async, sehingga kita perlu menunggu sampai proses Async nya selesai.
- Untuk mendapatkan informasi AJAX, kita bisa menggunakan AJAX Callback, yang akan dieksekusi setelah proses AJAX selesai

Membuat AJAX Callback

```
const ajax = new XMLHttpRequest();

ajax.onload = function(){
     const response = ajax.responseText;
};

ajax.open("METHOD", "url");
ajax.send()
```

Let's Code

https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/ajax.html

AJAX Error Callback

AJAX Error Callback

- AJAX adalah proses komunikasi Client dan Server
- Dalam komunikasi Client dan Server, kita tidak bisa selalu menganggap proses tersebut akan berjalan lancar.
- Akan ada banyak hal-hal yang bisa mengganggu proses AJAX yang bisa menyebabkan error, seperti; koneksi internet bermasalah, error dari server, data dari client tidak valid, dan lain-lain.
- Hal-hal error seperti ini perlu ditangani pada kode program kita. Dan kita bisa menggunakan Error Callback di AJAX

Membuat AJAX Callback

```
const ajax = new XMLHttpRequest();

ajax.onload = function(){
     if (ajax.status === 200){
          const response = ajax.responseText;
     } else {
          // error handler here
     }
};
```

Let's Code

https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/ajax.html

Dynamic Callback

Dynamic Callback

- Kadang dalam membuat program JavaScript, kita ingin membuat callback yang dinamis.
- Bisa berubah-ubah sesuai kebutuhan kita
- Untuk membuat Dynamic Callback, kita bisa memanfaatkan function as argument di JavaScript, dimana callback function nya kita masukkan dalam argument, sehingga bisa diubah sesuai dengan keinginan kita

Let's Code

https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/ajax.html

Promise

Masalah dengan Callback

```
doFirst(data, function () {
     doSecond(data, function () {
          doThird(data, function () {
               // Callback Hell
          })
     })
})
```

Apa itu Promise?

- Promise merupakan proxy untuk sebuah nilai di masa depan (Future) yang belum diketahui saat pembuatan Promise tersebut.
- Biasa Promise digunakan sebagai proxy untuk proses Async.
- Penggunaan Promise sangat mudah, dan lebih mirip dengan kode Synchronous.

Promise State

state	pending	fulfilled	rejected
result	undefined	value	error

Membuat Promise

```
const promise = new Promise(function (resolve, reject) {
    if(success){
        resolve(value)
    }else{
        reject(error)
    }
})
```

Let's Code

https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/promise.html

Promise Then Method

Promise Then Method

- Pertanyaannya, bagaimana cara mendapatkan value yang ada di Promise ketika value nya sudah ada?
- Promise memiliki method yang bernama then. Then method ini bisa digunakan sebagai callback ketika value pada Promise telah di resolve.
- Yang menarik menggunakan Then Method adalah, kita bisa membuat chain method, sehingga tidak akan terjebak pada Callback Hell

Menggunakan Then pada Promise

```
promise
     .then(function(value){
          // do something here
          return otherValue;
     })
     .then(function(otherValue){
          // do something here
          return otherValueAgain;
     });
```

Let's Code

https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/promise.html

Promise Catch Method

Promise Catch Method

- Pada AJAX, jika terjadi error, kita bisa menggunakan Error Callback, bagaimana dengan Promise?
- Promise memiliki method yang bernama Catch. Catch Method ini digunakan sebagai Error Callback yang bisa gunakan seperti Then Method.

Menggunakan Catch pada Promise

```
promise
     .then(function(value){
          // do something here
          return otherValue;
     })
     .catch(function(error){
          // do something here
     });
```

Let's Code

https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/promise.html

Promise Finally Method

Promise Finally Method

- Kadang kita ingin menjalankan kode tertentu, baik itu saat sukses ataupun error.
- Hal ini bisa dilakukan juga di Promise, menggunakan Finally Method

Menggunakan Finally pada Promise

```
promise
     .then(function(value){
          // do something here
          return otherValue;
     })
     .finally(function(){
          // do something here
     });
```

Let's Code

https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/promise.html

Promise All Method

Promise All Method

- Kadang kita perlu berhadapan dengan beberapa proses Async sekaligus.
- Misal, mengambil detail data produk dari Server pada satu halaman web, dimana satu halaman bisa menampilkan lebih dari satu produk.
- Menggunakan Promise satu per satu sangatlah menyulitkan jika terlalu banyak, tapi untungnya Promisa memiliki method All.
- All method bisa kita gunakan untuk menggabungkan beberapa Promise, menjadi Promise baru yang berisi data Array hasil Promise-Promise tersebut

Menggunakan All pada Promise

```
Promise.all([promise1, promise2, promise...])
     .then(function(values){
          // do something with values
     });
```

Let's Code

https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/promise.html

Fetch API

Fetch API

- Fetch API adalah API baru untuk melakukan AJAX
- Tidak seperti AJAX yang menggunakan Callback, Fetch API menggunakan Promise, sehingga kita bisa mudah menggunakan Fetch API dibanding AJAX
- https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

Menggunakan Fetch API

```
fetch(url, config)
     .then(function(response){
          // do something here
     })
     .catch(function(error){
          // do something here
     });
```

Let's Code!

https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/fetch.html

Async Await

Async Await

- Async Await adalah fitur baru JavaScript yang digunakan untuk mempermudah proses pembuatan code Promise.
- Dengan menggunakan Async Await, kita bisa membuat kode Asynchronous dengan gaya Synchronous
- Async digunakan untuk menandakan bahwa Function tersebut adalah Async, dan mengembalikan Promise
- Await digunakan untuk mendapatkan value hasil dari Function yang mengembalikan Promise.
- Await hanya bisa digunakan dalam Async Function

Menggunakan Async Await

```
async function onSearch(keyword){
    const products = await searchProducts(keyword);

    clearProducts();
    displayProducts(products);
}
```

Let's Code!

https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/async-await.html

Async Await Error Handler

Async Await Error Handler

- Pada Callback dan Promise, ada mekanisme Error Handler yang bisa dilakukan. Bagaimana dengan Async Await?
- Pada Async Await, kita bisa menggunakan gara Synchronous untuk menggunakan Error Handler nya, yaitu menggunakan try-catch dan try-catch-finally

Async Await Error Handler

```
async function onSearch(keyword){
     try{
          const products = await searchProducts(keyword);
          clearProducts();
          displayProducts(products);
     }catch(error){
     }finally{
     }
}
```

Let's Code!

https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/async-await.html

Web Worker

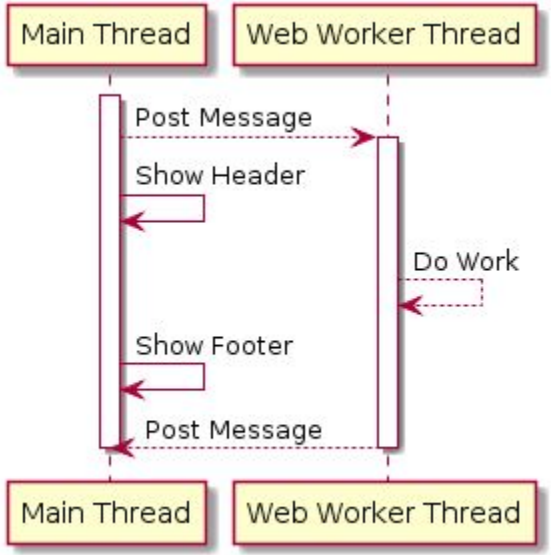
Sebelum Belajar Web Worker

- JavaScript adalah Single Thread, artinya walaupun proses yang kita buat adalah Async, tapi tetap akan dijalankan dalam Thread yang sama.
- Kemampuan satu Thread dalam mengelola beberapa pekerjaan, dinamakan Concurrent.
- Kemampuan menjalankan beberapa Thread untuk mengelola satu atau lebih pekerjaan, dinamakan Paralel.
- Dan untuk membuat proses secara Paralel, kita bisa menggunakan Web Worker

Web Worker

- Web Worker adalah kemampuan yang untuk menjalankan proses di Thread yang berbeda dibanding Main Thread.
- Keuntungan menggunakan Web Worker adalah, jika terdapat proses yang membutuhkan waktu lama, Web kita tidak akan Freeze, karena proses tersebut bisa kita jalankan di Thread yang berbeda dari Main Thread (yang biasa digunakan oleh UI)
- https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API

Web Worker


Main Thread
Web Worker Thread
Post Message
Show Header
Do Work
Show Footer
Post Message
Main Thread
Web Worker Thread

Pelajari :

Concurrency VS Parallelism

Membuat Web Worker

```
const worker = new Worker(“file.js”);
```

Let's Code!

https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/web-worker.html

Web Worker Communication

Web Worker Communication

- Web Worker adalah proses Async, dan untuk berkomunikasi dengan Web Worker, kita akan menggunakan Event Listener
- Untuk mengirim data ke Web Worker atau ke Main Thread, kita bisa menggunakan method postMessage

Web Worker Communication (1)

```
const worker = new Worker(“file.js”);

worker.addEventListener("message", function (event) {
     const data = event.data;
})

worker.postMessage(message)
```

Web Worker Communication (2)

```
// worker-file.js

addEventListener("message", function (event) {
    const data = event.data;

    // send back
    postMessage(message)
});
```

Let's Code!

https://github.com/khannedy/belajar-javascript-async/blob/master/code/before/web-worker.html

Selanjutnya?

Selanjutnya Belajar Apa?

- RxJS (Reactive Extensions Library for JavaScript)
- Web Socket
- Cara Kerja Non-Blocking

Keep Learning

Eko Kurniawan Khannedy

- Telegram : @khannedy
- Facebook : fb.com/khannedy
- Twitter : twitter.com/khannedy
- Instagram : instagram.com/programmerzamannow
- Email : echo.khannedy@gmail.com