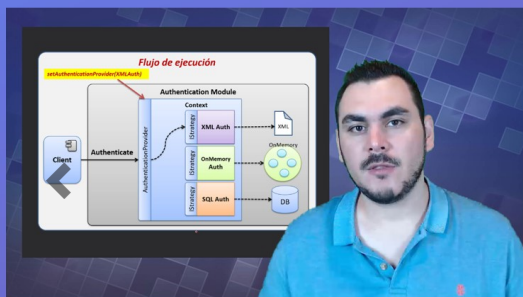


(.carousel)

(.carousel)



Ya disponible mi nueva plataforma de cursos

[VER CURSOS](#)

○○○

<https://centripio.io>

Javascript – Async/Await

📅 15 marzo, 2019 (<https://www.oscarblancarteblog.com/2019/03/15/javascript-async-await/>) 👤 oblancarte
(<https://www.oscarblancarteblog.com/author/oblancarte/>) 📁 o



La finalidad de los operadores `async` y `await` es simplificar aun más la forma en que trabajamos con las promesas, de tal forma que permite ejecutarlas y esperar el resultado de forma síncrona, si la necesidad de los famosos bloques `then` y `catch`.

El contenido de este artículo también lo explico por Youtube, recuerda suscribirte por que estaremos subiendo más contenido como este:

Javascript | Los operadores async/await



Los que ya tiene tiempo utilizando Javascript, recordaran lo complicado que era trabajar con las Callback y el famoso problema llamado callback hell, por suerte, luego apetecieron las promesas, una nueva forma de resolver el problema de asincronicidad de Javascript, permitiendo definir una serie de bloques `then` y `catch` que al final eran son muy parecidas a las callback, con la diferencia de que permitía tener una mejor organización del código, sin embargo, y pesar de estas mejoras significativas, Javascript seguía teniendo el mismo problema, y era la asincronicidad.

Dicho lo anterior, los operadores `async/await` se agregan a Javascript a partir de la versión ECMAScript 7 para simplificar la forma de trabajar con las promesas, con las cuales es posible ejecutarlas de forma síncrona y bloqueando la ejecución hasta que sean resueltas.

Para comprender mejor como funcionan los operadores `async / await` vamos a realizar un ejemplo, en el cual consumiremos un recurso de Internet mediante el API Fetch. En este primer ejemplo veremos como se hace sin `async / await` :

```
1  const fetch = require( 'node-fetch' )
2
3  function getCountry(){
4      return fetch('https://pkgstore.datahub.io/core/country-codes/country-codes_
5
6  }
7
8  let hello = getCountry()
9  hello.then(response => response.json())
10     .then(response => response.map(country => country['CLDR display name']))
11     .then(response => console.log(response))
```

Cómo podemos observar, hemos creado la función `getCountry` encargada de retornar una promesa con la búsqueda de un recurso de Internet, este método es ejecutado y seguido es necesario resolver la promesa con una serie de bloques `then`. Podemos observar que esto es un poco verbose, pues en cada bloque hay que definir la variable `response`, definir una callback (o arrow function) y retornar los resultados para ser procesados por el siguiente bloque `then`. Otro de los problemas es que el resultado solo estará disponible dentro del bloque `then`, complicando la forma en que trabajamos y manejamos los errores.

()



(<https://reactiveprogramming.io/books/aplicaciones-reactivas-con-react-nodejs-mongodb>)

¿Quieres aprender las tecnologías más importantes de la web? te invito a que veas mi libro donde aprenderás React, NodeJS y MongoDB y conviértete en un desarrollador FullStack de JavaScript ()

Ahora bien, antes de explicar como funciona `async / await` quiere que veas un ejemplo, lo análisis y después pasaremos a explicar todo a detalle:

```
1  const fetch = require( 'node-fetch' )
2
3  async function getCountry(){
4      let response = await fetch('https://pkgstore.datahub.io/core/country-codes/
5      let json = await response.json()
6      return json.map(country => country['CLDR display name'])
7  }
8
9  (async function(){
10     let hello = await getCountryAsync()
11     console.log("log => ", hello)
12 })()
```

Para empezar, vemos un código mucho más limpio, además, la función `getCountry` ejecuta todas las instrucciones de forma asíncrona. Pero que está pasado.

Lo primero que debemos de saber es que el operador `await` esperará hasta que la promesa sea resuelta, lo que provocará que el hilo de ejecución hasta que la promesa se resuelva, y una vez resuelta, el hilo de ejecución continuará donde se quedo.

Otra cosa importante, es que `await` solo se puede utilizar en funciones que tengan el operador `async`, en caso contrario, se lanzará el siguiente error:

```
SyntaxError: await is only valid in async function
```

Los método con el operadores `async` son llamados `async methods`, y no solo permiten utilizar el operador `async`, si no que provocara que todo lo que retornemos sea encapsulado dentro de una promesa:

```
1  async function helloWorld(){
2      return "hello world"
3  }
4  let hello = helloWorld()
5  console.log(hello)
6
7  // Output
8  // Promise { 'hello world' }
```

En este ejemplo tenemos una función que solo regresa un string, sin embargo, vemos que en el output, el string "hello world" está encapsulado dentro de una promesa.

Regresando al ejemplo, si ejecutamos la función `getCountry`, este nos regresará un promesa, por lo que si imprimiéramos el resultado de la función podemos observar que nos regresa una promesa

```
1 console.log(getCountry())
2
3 // Output
4 // Promise { <pending> }
```

En este punto te podrías estar preguntando, ¿que sentido tiene utilizar `await`, si al final me regresará una promesa?, puede que tengas razón sin embargo, recordemos que con `await` podemos esperar hasta que se resuelva una promesa, entonces podemos hacer lo siguiente:

```
1 (async function(){
2     let hello = await getCountry()
3     console.log("log => ", hello)
4 })
5
6 // Output
7 // 'Guinea-Bissau',
8 // 'Guyana',
9 // 'Haiti',
10 // ... 150 more items ]
```

En este ejemplo, vemos que hemos ejecutado la función `getCountry` dentro de una `async method` por lo que podríamos utilizar `await` para resolver la respuesta de `getCountry`.

Excepciones


Otra de las grandes ventajas que ofrece el operadores `async / await` es que nos permite controlar las excepciones mediante el clásico bloque `try-catch`, sin necesidad de utilizar bloque `.catch()` de las promesas:


```
1  async function getCountry(){
2      try {
3          let response = await fetch('https://pkgstore.datahub.io/core/country-cc
4          let json = await response.json()
5          return json.map(country => country['CLDR display name'])
6      } catch (err) {
7          console.log("Error ==> ", err)
8      }
9  }
```


Conclusiones


Cómo hemos podido comprobar, los operadores `async / await` proporcionan una forma mucho más imple para trabajar con promesas, permitiendo trabajar de forma síncrona.

Compártelo:

 LinkedIn (<https://www.oscarblancarteblog.com/2019/03/15/javascript-async-await/?share=linkedin&nb=1>)

 Facebook (<https://www.oscarblancarteblog.com/2019/03/15/javascript-async-await/?share=facebook&nb=1>)

 Twitter (<https://www.oscarblancarteblog.com/2019/03/15/javascript-async-await/?share=twitter&nb=1>)

 WhatsApp (<https://api.whatsapp.com/send?text=Javascript%20-%20Async%2FAwait%20https%3A%2F%2Fwww.oscarblancarteblog.com%2F2019%2F03%2F15%2Fjavascript-async-await%2F>)

Relacionado

 Introducción a NodeJS

(JavaScript del lado del Servidor)

(<https://www.oscarblancarteblog.com/2017/05/29/introduccion-a-nodejs-2/>)

Introducción a NodeJS (JavaScript del lado del Servidor)

(<https://www.oscarblancarteblog.com/2017/05/29/introduccion-a-nodejs-2/>)



(<https://www.oscarblancarteblog.com/2017/03/02/introduccion-a-webworkers/>)

Introducción a WebWorkers (<https://www.oscarblancarteblog.com/2017/03/02/introduccion-a-webworkers/>)



(<https://www.oscarblancarteblog.com/2016/11/22/html5-antecedentes-novedades/>)
HTML5, antecedentes y novedades
(<https://www.oscarblancarteblog.com/2016/11/22/html5-antecedentes-novedades/>)

og.com/2017/05/29/introducci
on-a-nodejs-2/)

og.com/2017/03/02/introducci
on-a-webworkers/)

og.com/2016/11/22/html5-
antecedentes-novedades/)

javascript (<https://www.oscarblancarteblog.com/tag/javascript/>)

DEJA UN COMENTARIO

Tu dirección de correo electrónico no será publicada. Los campos obligatorios están marcados con *

Comentario

Nombre *

Correo electrónico *

Web

☐ Recibir un email con los siguientes comentarios a esta entrada.

☐ Recibir un email con cada nueva entrada.

PUBLICAR COMENTARIO

◀ Javascript – la función filter (<https://www.oscarblancarteblog.com/2019/03/12/javascript-la-funcion-filter/>)

Netflix – Chaos Monkey ▶ (<https://www.oscarblancarteblog.com/2019/03/18/netflix-chaos-monkey/>)

MIS CURSOS AL 50%

 Centripio (<https://centripio.io/>)

...



SUSCRÍBETE AL BLOG POR CORREO ELECTRÓNICO

Introduce tu correo electrónico para suscribirte a este blog y recibir notificaciones de nuevas entradas.

Únete a otros 1.009 suscriptores

Dirección de correo electrónico

SUSCRIBIR

HOLA, SOY OSCAR BLANCARTE



Autor, Software Architect & Full Stack Developer con más de 14 años de experiencia en el desarrollo de software y en la industria de las tecnologías de la información. Apasionado por las tecnologías y el software en general.

SÍGUEME EN TWITTER

Seguir a @oscarjblancarte

APLICACIONES REACTIVAS CON REACT, NODEJS & MONGODB



(<https://reactiveprogramming.io>)

MI NUEVO LIBRO: INTRODUCCIÓN A LOS PATRONES DE DISEÑO – UN ENFOQUE PRÁCTICO



(<https://reactiveprogramming.io/books/design-patterns/es>)

TAMBIEN ESTOY EN:



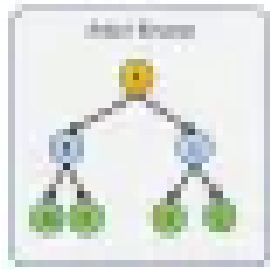
MI CANAL DE YOUTUBE



([https://www.youtube.com/centripio?](https://www.youtube.com/centripio?sub_confirmation=1)

[sub_confirmation=1\)](https://www.youtube.com/centripio?sub_confirmation=1)

ARTICULOS POPULARES



(<https://www.oscarblancarteblog.com/2014/08/22/estructura-de-datos-arboles/>)

Estructura de datos - Árboles

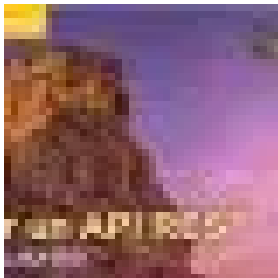
(<https://www.oscarblancarteblog.com/2014/08/22/estructura-de-datos-arboles/>)



(<https://www.oscarblancarteblog.com/2017/03/06/soap-vs-rest-2/>)

SOAP vs REST ¿cual es mejor?

(<https://www.oscarblancarteblog.com/2017/03/06/soap-vs-rest-2/>)



(<https://www.oscarblancarteblog.com/2018/06/25/creando-un-api-rest-en-java-parte-1/>)

Creando un API REST en Java (parte 1)

(<https://www.oscarblancarteblog.com/2018/06/25/creando-un-api-rest-en-java-parte-1/>)



(<https://www.oscarblancarteblog.com/2018/07/17/spring-boot-relacion-los-microservicios/>)

Que es Spring Boot y su relación con los microservicios

(<https://www.oscarblancarteblog.com/2018/07/17/spring-boot-relacion-los-microservicios/>)

(<https://www.oscarblancarteblog.com/2018/11/30/data-transfer-object-dto-patron-diseno/>)



Data Transfer Object (DTO) – Patrón de diseño

(<https://www.oscarblancarteblog.com/2018/11/30/data-transfer-object-dto-patron-diseno/>)



(<https://www.oscarblancarteblog.com/2018/12/03/metodos-http-rest/>)

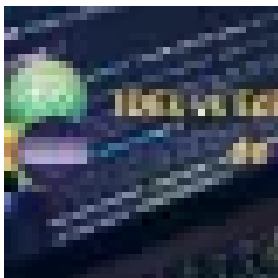
Métodos HTTP (REST)

(<https://www.oscarblancarteblog.com/2018/12/03/metodos-http-rest/>)



(<https://www.oscarblancarteblog.com/2016/12/19/web-services-con-java-jax-ws/>) Web Services con Java (JAX-WS)

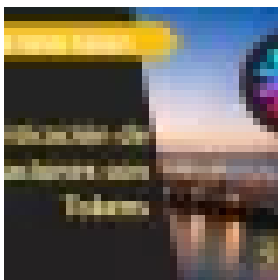
(<https://www.oscarblancarteblog.com/2016/12/19/web-services-con-java-jax-ws/>)



(<https://www.oscarblancarteblog.com/2017/10/26/ide-vs-editor-de-texto/>)

IDE vs Editor de texto

(<https://www.oscarblancarteblog.com/2017/10/26/ide-vs-editor-de-texto/>)



(<https://www.oscarblancarteblog.com/2017/06/08/autenticacion-con-json-web-tokens/>)

Autenticación con JSON Web Tokens

(<https://www.oscarblancarteblog.com/2017/06/08/autenticacion-con-json-web-tokens/>)

TEMAS

2phase commite (<https://www.oscarblancarteblog.com/tag/2phase-commite/>)

API REST (<https://www.oscarblancarteblog.com/tag/api-rest/>)

Arboles (<https://www.oscarblancarteblog.com/tag/arboles/>)

[arquitectura](https://www.oscarblancarteblog.com/tag/arquitectura/) (<https://www.oscarblancarteblog.com/tag/arquitectura/>)

[Arquitectura de software](https://www.oscarblancarteblog.com/tag/arquitectura-de-software/) (<https://www.oscarblancarteblog.com/tag/arquitectura-de-software/>)

[BaaS](https://www.oscarblancarteblog.com/tag/baas/) (<https://www.oscarblancarteblog.com/tag/baas/>)

[Base de datos](https://www.oscarblancarteblog.com/tag/base-de-datos/) (<https://www.oscarblancarteblog.com/tag/base-de-datos/>)

[bpel](https://www.oscarblancarteblog.com/tag/bpel-2/) (<https://www.oscarblancarteblog.com/tag/bpel-2/>)

[centripio](https://www.oscarblancarteblog.com/tag/centripio/) (<https://www.oscarblancarteblog.com/tag/centripio/>)

[certificaciones](https://www.oscarblancarteblog.com/tag/certificaciones/) (<https://www.oscarblancarteblog.com/tag/certificaciones/>)

[composite](https://www.oscarblancarteblog.com/tag/composite/) (<https://www.oscarblancarteblog.com/tag/composite/>)

[desarrollo web](https://www.oscarblancarteblog.com/tag/desarrollo-web/) (<https://www.oscarblancarteblog.com/tag/desarrollo-web/>)

[diseño](https://www.oscarblancarteblog.com/tag/disenio/) (<https://www.oscarblancarteblog.com/tag/disenio/>)

[Estructura de datos](https://www.oscarblancarteblog.com/tag/estructura-de-datos/) (<https://www.oscarblancarteblog.com/tag/estructura-de-datos/>)

[express](https://www.oscarblancarteblog.com/tag/express/) (<https://www.oscarblancarteblog.com/tag/express/>)

[hibernate](https://www.oscarblancarteblog.com/tag/hibernate-2/) (<https://www.oscarblancarteblog.com/tag/hibernate-2/>)

[HTML](https://www.oscarblancarteblog.com/tag/html/) (<https://www.oscarblancarteblog.com/tag/html/>) [java](https://www.oscarblancarteblog.com/tag/java-2/) (<https://www.oscarblancarteblog.com/tag/java-2/>)

[java8](https://www.oscarblancarteblog.com/tag/java8/) (<https://www.oscarblancarteblog.com/tag/java8/>)

[javascript](https://www.oscarblancarteblog.com/tag/javascript/) (<https://www.oscarblancarteblog.com/tag/javascript/>)

[JAX-RS](https://www.oscarblancarteblog.com/tag/jax-rs/) (<https://www.oscarblancarteblog.com/tag/jax-rs/>) [JMS](https://www.oscarblancarteblog.com/tag/jms/) (<https://www.oscarblancarteblog.com/tag/jms/>)

[jpa](https://www.oscarblancarteblog.com/tag/jpa-2/) (<https://www.oscarblancarteblog.com/tag/jpa-2/>)

[Marca personal](https://www.oscarblancarteblog.com/tag/marca-personal/) (<https://www.oscarblancarteblog.com/tag/marca-personal/>)

[microservicios](https://www.oscarblancarteblog.com/tag/microservicios/) (<https://www.oscarblancarteblog.com/tag/microservicios/>)

[mongodb](https://www.oscarblancarteblog.com/tag/mongodb/) (<https://www.oscarblancarteblog.com/tag/mongodb/>)

[NodeJS](https://www.oscarblancarteblog.com/tag/nodejs/) (<https://www.oscarblancarteblog.com/tag/nodejs/>)

[oracle](https://www.oscarblancarteblog.com/tag/oracle/) (<https://www.oscarblancarteblog.com/tag/oracle/>)

[Patron de diseño](https://www.oscarblancarteblog.com/tag/patron-de-diseno/) (<https://www.oscarblancarteblog.com/tag/patron-de-diseno/>)

[Patrones de diseño](https://www.oscarblancarteblog.com/tag/patrones-de-diseno/) (<https://www.oscarblancarteblog.com/tag/patrones-de-diseno/>)

[POO](https://www.oscarblancarteblog.com/tag/poo/) (<https://www.oscarblancarteblog.com/tag/poo/>)

[Programación orientada a objetos](https://www.oscarblancarteblog.com/tag/programacion-orientada-a-objetos/) (<https://www.oscarblancarteblog.com/tag/programacion-orientada-a-objetos/>)

[proxy](https://www.oscarblancarteblog.com/tag/proxy/) (<https://www.oscarblancarteblog.com/tag/proxy/>) [react](https://www.oscarblancarteblog.com/tag/react/) (<https://www.oscarblancarteblog.com/tag/react/>)

[Schema](https://www.oscarblancarteblog.com/tag/schema/) (<https://www.oscarblancarteblog.com/tag/schema/>)

[Seguridad](https://www.oscarblancarteblog.com/tag/seguridad/) (<https://www.oscarblancarteblog.com/tag/seguridad/>)

[soa](https://www.oscarblancarteblog.com/tag/soa/) (<https://www.oscarblancarteblog.com/tag/soa/>)

[soa suite](https://www.oscarblancarteblog.com/tag/soa-suite-2/) (<https://www.oscarblancarteblog.com/tag/soa-suite-2/>)

[software](https://www.oscarblancarteblog.com/tag/software/) (<https://www.oscarblancarteblog.com/tag/software/>)

[spring boot](https://www.oscarblancarteblog.com/tag/spring-boot/) (<https://www.oscarblancarteblog.com/tag/spring-boot/>)

[Weblogic](https://www.oscarblancarteblog.com/tag/weblogic/) (<https://www.oscarblancarteblog.com/tag/weblogic/>)

[webservice](https://www.oscarblancarteblog.com/tag/webservice/) (<https://www.oscarblancarteblog.com/tag/webservice/>)

[WebSocket](https://www.oscarblancarteblog.com/tag/websocket/) (<https://www.oscarblancarteblog.com/tag/websocket/>)

[XML](https://www.oscarblancarteblog.com/tag/xml/) (<https://www.oscarblancarteblog.com/tag/xml/>) [XSD](https://www.oscarblancarteblog.com/tag/xsd/) (<https://www.oscarblancarteblog.com/tag/xsd/>)

SÍGUEME EN TWITTER



Oscar Blancarte

@oscarjblancarte

¡Oscar!, inyectamos un bug a producción, pero no lo ha visto el cliente, ¿Que hacemos?

-YO:



10 dic. 2019



Oscar Blancarte

@oscarjblancarte

Para cuando los String templates para Java como los tiene JavaScript y que además internamente usen StringBuilder 🤔

6 dic. 2019

Oscar Blancarte retweeteó



centripio

@centripio

Te gustaría conocer mas sobre los asistentes

inteligentes?, aquí les compartimos un artículo :

centripio.io/blog/crea-tu-p... ➡

3 dic. 2019

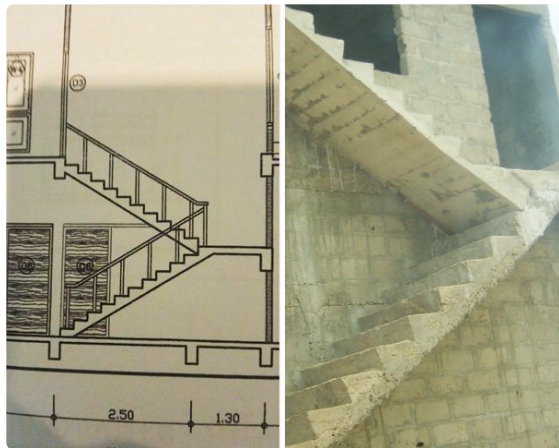


Oscar Blancarte

@oscarjblancarte



Diagrama de arquitectura VS 😞 implementación



1 dic. 2019



Oscar Blancarte

@oscarjblancarte



El Black friday es literalmente el Black friday
para los programadores

30 nov. 2019