

Axel Sánchez

148781

PROYECTO INTEGRAL WEB

JESUS ROBERTO SAUCEDO FRANCO



UACJ

Async/Await y Promises en Javascript

Promises

Las Promesas de JavaScript fueron introducidas con ECMAScript 6, y trajeron al lenguaje una forma más concisa de trabajar con funciones asíncronas.

Las Promesas de JavaScript reciben una función con dos parámetros (resolve, reject) que son funciones que se llaman una vez el proceso se ejecute con éxito (resolve) o falle (reject).

Las promesas poseen tres estados:

- Pending (pendiente, esperando ser resuelta o rechazada)
- Fulfilled (resuelta)
- Rejected (rechazada)

Las Promesas de JavaScript nos dan mayor flexibilidad a la hora de manejar código asíncrono. Uno de los usos más comunes de las promesas en el desarrollo web, es poder controlar el flujo de peticiones a servidores web ya que, dependiendo de la conexión y otros factores, el tiempo exacto que demora una respuesta es impredecible.

Async/Await

Async/Await solo funcionan dentro de funciones asíncronas, es decir, las funciones con la sintaxis `async function nombre_de_funcion () {}`

Cuando se llama a una función `async`, esta devuelve un elemento `Promise`. Cuando la función `async` devuelve un valor, `Promise` se resolverá con el valor devuelto. Si la función `async` genera una excepción o algún valor, `Promise` se rechazará con el valor generado.

Una función `async` puede contener una expresión `await`, la cual pausa la ejecución de la función asíncrona y espera la resolución de la `Promise` pasada y, a continuación, reanuda la ejecución de la función `async` y devuelve el valor resuelto.

<https://blog.nexlab.dev/tech/2018/08/24/js-promises-para-codigo-asincrono.html>

<https://blog.nexlab.dev/tech/2018/08/31/que-es-async-await-en-javascript.html#asyncawait>

https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias/funcion_asincrona