Universidad Autónoma de Ciudad Juárez
Instituto de Arquitectura, Diseño y Arte
Departamento de Diseño
Licenciatura en Diseño Digital de Medios Interactivos

Async / Await

Ericka Sánchez Corral

Proyecto integral web

Ciudad Juárez, Chihuahua, febrero 2020
UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ
INSTITUTO DE ARQUITECTURA, DISEÑO Y ARTE
Departamento de Diseño

Tarea:

Async / Awayt (Promise)

Definición:

Async / Await es un tipo de promesa. Las promesas en JavaScript son objetos que pueden tener múltiples estados, esto lo hacen porque lo que se solicita no está disponible de inmediato y se requiere de poder detectar en qué estado se encuentra (Drasner, 2020).

Async:

Las funciones asíncronas permiten escribir código basado en promesas como si fuera síncrono, pero sin bloquear el hilo de ejecución. Funciona de forma asíncrona a través del bucle de eventos. Las funciones asíncronas siempre devolverán un valor. El uso *async* simplemente implica que se devolverá una promesa, y si *promise* no se devuelve, JavaScript lo envuelve automáticamente en una resolución *promise* con su valor (Mandwarva, 2019).

Await:

El operador de espera se utiliza para esperar una Promesa. Se puede usar solo dentro de un bloque *async.* La palabra clave *await* hace que JavaScript espere hasta que la promesa devuelva un resultado. Cabe señalar que solo hace que el *async* bloque de funciones espere y no toda la ejecución del programa (Mandwarya, 2019).

De acuerdo con MDN:

los Promise.all() El método devuelve un solo Promiseque se resuelve cuando se han resuelto todas las promesas aprobadas como iterables o cuando el iterable no contiene promesas. Rechaza con la razón de la primera promesa que rechaza (MDN, S.F.).

Estados:

1. **Pendiente:** llama por primera vez una promesa y no se sabe qué devolverá.
2. **Cumplido:** la operación se ha completado con éxito.
3. **Rechazado:** la operación falla.

Ejemplos:

JavaScript

```
const getSomeTacos = new Promise((resolve, reject) => {
  console.log("Initial state: Excuse me can I have some tacos");

  resolve();
})
  .then(() => {
    console.log("Order some tacos");
  })
  .then(() => {
    console.log("Here are your tacos");
  })
  .catch(err => {
    console.error("Nope! No tacos for you.");
  });
```

console

```
> Initial state: Excuse me can I have some tacos
> Order some tacos
> Here are your tacos
```

JavaScript

```
const getSomeTacos = new Promise((resolve, reject) => {
  console.log("Initial state: Excuse me can I have some tacos");

  reject();
})
  .then(() => {
    console.log("Order some tacos");
  })
  .then(() => {
    console.log("Here are your tacos");
  })
  .catch(err => {
    console.error("Nope! No tacos for you.");
  });
```

console

```
> Initial state: Excuse me can I have some tacos
> Nope! No tacos for you.
```

```
// this is the function we want to schedule. it's a promise.
const addOne = (x) => {
  return new Promise(resolve => {
    setTimeout(() => {
      console.log(`I added one! Now it's ${x + 1}.`)
      resolve()
    }, 2000);
  })
}

// we will immediately log the first one,
// then the addOne promise will run, taking 2 seconds
// then the final console.log will fire
async function addAsync() {
  console.log('I have 10')
  await addOne(10)
  console.log(`Now I'm done!`)
}

addAsync()
```

```
console
> I have 10
> I added one! Now it's 11.
> Now I'm done!
```

BIBLIOGRAFÍA

Drasner, S. (2020). CSS Tricks. *Comprensión Async Await.* Recuperado de https://css-tricks.com/understanding-async-await/

Mandwarva, A. (2019). Medium. *Cómo usar Async Await en JavaScript.* Recuperado de https://medium.com/javascript-in-plain-english/async-await-javascript-5038668ec6eb

MDN. (S.F.). MDN Web Docs. *Async Function.* Recuperado de https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function